НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра електронних комунікацій та інтернету речей

«На правах рукопису»
УДК 621.39

До захисту допущено:
Завідувач кафедри
_____ Анатолій МАКАРЕНКО
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системи електронних комунікацій та Інтернету речей»

зі спеціальності 172 «Електронні комунікації та радіотехніка»

на тему: «Протоколи рівня застосунків в архітектурі Інтернету речей: дослідження ефективності та особливостей»

Виконала:
студентка ІІ курсу, групи ЦС-41мп
Дідковська Наталія Андріївна _____

Науковий керівник:
Доцент каф. ЕКІР к.т.н.
Григоренко Олена Григорівна _____

Рецензент:
Доцент каф. Інформаційної безпеки НН ФТІ к.т.н.
Репа Федір Михайлович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут телекомунікаційних систем
Кафедра електронних комунікацій та інтернету речей

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Електронні комунікації та радіотехніка»

Освітньо-професійна програма «Системи електронних комунікацій та Інтернету речей»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Анатолій МАКАРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Дідковській Наталії Андріївні

1. Тема дисертації «Протоколи рівня застосунків в архітектурі Інтернету речей: дослідження ефективності та особливостей», науковий керівник дисертації Григоренко Олена Григорівна, Доцент каф. ЕКІР к.т.н., затверджені наказом по університету від «03» листопада 2025 р. №4772-с.

2. Термін подання студентом дисертації 18.12.2025

3. Об’єкт дослідження: процес обміну даними між компонентами розподілених систем Інтернету речей.

4. Вихідні дані: наукові статті, рекомендації ITU-T, підручники за темою технології IoT.

5. Перелік завдань, які потрібно розробити:

1. Теоретичні основи Інтернету речей
2. Архітектурні моделі IoT
3. Протоколи рівня застосунків в IoT: структура, класифікація та аналіз
4. Дослідження та практична реалізація протоколів рівня застосунків

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: електронна презентація PowerPoint:

Слайд № 1 Титульний слайд

Слайд № 2 Мета та актуальність роботи

Слайд № 3 Сутність та базові принципи функціонування Інтернету речей

Слайд № 4 Архітектурні моделі IoT

Слайд № 5 Архітектурні моделі IoT

Слайд № 6 Класифікація та функціональні особливості основних протоколів прикладного рівня

Слайд № 7 Протокол MQTT

Слайд № 8 Протокол DDS

Слайд № 9 Протокол CoAP

Слайд № 10 Протокол AMQP

Слайд № 11 Протокол HTTP

Слайд № 12 Порівняльний аналіз протоколів

Слайд № 13 Постановка задачі та методика дослідження

Слайд № 14 Приклад реалізації та аналіз роботи протоколів

Слайд № 15 Приклад реалізації та аналіз роботи протоколів

Слайд № 16 Результати дослідження

Слайд № 17 Рекомендації щодо вибору протоколу для різних типів IoT-рішень

Слайд № 18 Висновки

7. Дата видачі завдання 01.09.2025

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення літератури та стандартів для написання магістерської дисертації	01.09.2025-10.09.2025	Виконано
2.	Написання змісту та вступу до дисертації	10.09.2025-12.09.2025	Виконано
3.	Написання першого розділу: Теоретичні основи Інтернету речей	12.09.2025-30.09.2025	Виконано
4.	Написання другого розділу: Протоколи рівня застосунків в IoT: структура, класифікація та аналіз	01.10.2025-31.10.2025	Виконано
5.	Написання третього розділу: Дослідження та практична реалізація обміну даними	01.11.2025-15.11.2025	Виконано

	протоколів рівня застосунків		
6.	Проведення дослідження	15.11.2025-28.11.2025	Виконано
7.	Написання висновків	29.11.2025-30.11.2025	Виконано
8.	Оформлення роботи	01.12.2025-07.12.2025	Виконано
9.	Розробка доповіді та презентації	08.12.2025-14.12.2025	Виконано

Студент Наталія ДІДКОВСЬКА

Науковий керівник Олена ГРИГОРЕНКО

РЕФЕРАТ

Темою магістерської дисертації є дослідження протоколів рівня застосунків в архітектурі Інтернету речей: їх ефективності та особливостей.

Робота містить 89 сторінок, зокрема 34 ілюстрації, 6 таблиць та 40 джерел інформації.

Тема магістерської дисертації є актуальною, оскільки зі зростанням масштабів впровадження рішень Інтернету речей у різних галузях підвищуються вимоги до надійності, енергоефективності, продуктивності та безпеки. У таких системах протоколи рівня застосунків відіграють ключову роль, оскільки саме вони визначають формат переданих повідомлень, модель взаємодії між вузлами, механізми підтвердження доставки та підтримку якості обслуговування (QoS), що безпосередньо впливає на функціонування IoT-платформ і прикладних сервісів.

Метою магістерської дисертації є комплексне дослідження протоколів рівня застосунків в архітектурі Інтернету речей з акцентом на аналізі їх технічних характеристик, порівнянні ефективності та визначенні умов оптимального використання. Досягнення мети передбачає теоретичний огляд, експериментальний аналіз та розгляд практичних прикладів застосування протоколів у сучасних IoT-системах.

У дисертації запропоновано методику комплексної оцінки протоколів рівня застосунків в архітектурі Інтернету речей, яка враховує як функціональні особливості протоколів, так і їх поведінку в умовах, наближених до реальних сценаріїв. На основі результатів експериментальних досліджень сформовано рекомендації щодо вибору протоколів прикладного рівня, що дозволяє підвищити ефективність, надійність та безпеку обміну даними в системах Інтернету речей.

КЛЮЧОВІ СЛОВА: IoT, Інтернет речей, архітектурні моделі, протокол, протоколи рівня застосунків, MQTT, DDS, CoAP, AMQP, HTTP, методика.

ABSTRACT

The topic of the master's thesis is the study of application-level protocols in the Internet of Things architecture: their efficiency and features.

The work contains 89 pages, including 34 illustrations, 6 tables, and 40 sources of information.

The topic of the master's thesis is relevant because, with the growing scale of implementation of Internet of Things solutions in various industries, the requirements for reliability, energy efficiency, performance, and security are increasing. In such systems, application layer protocols play a key role, as they determine the format of transmitted messages, the model of interaction between nodes, delivery confirmation mechanisms, and quality of service (QoS) support, which directly affects the functioning of IoT platforms and application services.

The aim of the master's thesis is to conduct a comprehensive study of application layer protocols in the Internet of Things architecture, with an emphasis on analysing their technical characteristics, comparing their effectiveness, and determining the conditions for their optimal use. Achieving this goal involves a theoretical review, experimental analysis, and consideration of practical examples of the application of protocols in modern IoT systems.

The thesis proposes a methodology for a comprehensive assessment of application-level protocols in the Internet of Things architecture, which takes into account both the functional features of the protocols and their behaviour in conditions close to real-life scenarios. Based on the results of experimental research, recommendations have been formulated for the selection of application-level protocols, which allows for increased efficiency, reliability, and security of data exchange in Internet of Things systems.

KEYWORDS: IoT, Internet of Things, architectural models, protocol, application layer protocols, MQTT, DDS, CoAP, AMQP, HTTP, methodology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ISO	International Organization for Standardization – Міжнародна організація зі стандартизації)
IoT	Internet of Things –Інтернет речей
QoS	Quality of Service – Рівень якості обслуговування
MQTT	Message Queuing Telemetry Transport – Протокол передачі телеметричних повідомлень через чергу
DDS	Data Distribution Service – Служба розподілу даних
CoAP	Constrained Application Protocol – Протокол обмеженого застосування
AMQP	Advanced Message Queuing Protocol - Розширений протокол черг повідомлень
HTTP	Hypertext Transfer Protocol – Протокол передачі гіпертексту
Wi-Fi	Wireless Fidelity – Бездротова локальна мережа
LPWAN	Low-Power Wide-Area Network – Мережа з низьким енергоспоживанням і великим радіусом дії
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector – Міжнародний союз електрозв'язку - Сектор стандартизації електрозв'язку
IP	Internet Protocol – Протокол Інтернету
IEEE	Institute of Electrical and Electronics Engineers - Інститут інженерів з електротехніки та електроніки
BLE	Bluetooth Low Energy – Bluetooth з низьким енергоспоживанням
RFID	Radio Frequency Identification - Ідентифікація за допомогою радіочастот
IIoT	Industrial Internet of Things – Промисловий Інтернет речей

NFC	Near Field Communication - Технологія ближнього радіозв’язку
PLC	Programmable Logic Controller - Програмований логічний контролер
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
IPv4/6	Internet Protocol version 4/6 – Інтернет-протокол версії 4
LoRaWAN	Long Range Wide Area Network - Мережевий протокол далекої дії з низьким енергоспоживанням
NB-IoT	Narrowband Internet of Things - Вузькосмуговий Інтернет речей
LTE-M	Long Term Evolution for Machines - Технологія LTE для машинного зв’язку (машин-тип комунікацій)
3G/4G/5G	третє, четверте та п’яте покоління мобільного стільникового зв’язку (G —generation)
CAN	Controller Area Network - мережа контролерів
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
VPN	Virtual Private Network — Віртуальна приватна мережа
API	Application Programming Interface - Прикладний програмний інтерфейс
DoS	Denial of Service – Відмова в обслуговуванні
NoSQL	Нереляційні бази даних
KPI	ключові показники ефективності
ERP	Enterprise Resource Planning - Планування ресурсів підприємства
CRM	Customer Relationship Management - Управління взаємовідносинами з клієнтами
BPM	Business Process Management - Управління бізнес-процесами
BI	Business Intelligence - Бізнес-аналітика
MCU	Microcontroller Unit - Мікроконтролерний модуль
STM32	32-бітний мікроконтролер

AVR	8-бітна RISC-архітектура мікроконтролерів
RTU/TCP	Remote Terminal Unit over TCP
OPC-UA	OPC Unified Architecture
VLAN	Virtual Local Area Network - віртуальна локальна мережа
ЦОД	Центр Обробки Даних (аналог англ. Data Center)
БД	База Даних (англ. Database)
VM	Virtual Machine - віртуальна машина
HMI	Human–Machine Interface - інтерфейс «людина–машина»
HAL	Hardware Abstraction Layer - рівень абстракції апаратного забезпечення
NAT	Network Address Translation - трансляція мережевих адрес
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
OTA	Over-the-Air
SCADA	Supervisory Control And Data Acquisition
HMI	Human–Machine Interface
CMMS	Computerized Maintenance Management System - Управління обслуговуванням обладнання
MitM	Man-in-the-Middle
M2M	Machine-to-Machine – Міжмашинна взаємодія
DLRL	Data Local Reconstruction Layer - Рівень локальної реконструкції даних
DCPS	Data-Centric Publish-Subscribe - Публікування-підписка, орієнтована на дані
WoT	WEB of Things – Веб Речей
REST	Representational State Transfer – Представлення переходу стану
TCP	Transport Control Protocol – Протокол керування передачею даних
DTLS	Datagram Transport Layer Security – Протокол забезпечення

	безпеки транспортного рівня датаграм
URL	Uniform Resource Locator - Універсальний локатор ресурсу
URI	Uniform Resource Identifier – Уніфікований ідентифікатор ресурсу

ВСТУП

Сучасний етап розвитку інформаційно-комунікаційних технологій характеризується переходом від ізольованих інформаційних систем до розподілених кіберфізичних комплексів, що інтегрують комунікаційні мережі та фізичні об’єкти. Однією з ключових технологічних парадигм цього переходу є Інтернет речей (Internet of Things, IoT), який забезпечує масове підключення сенсорів, виконавчих механізмів, вбудованих пристроїв і сервісів до глобального інформаційного простору. У таких умовах стрімко зростають обсяги трафіку, підвищуються вимоги до масштабованості, надійності, оперативності та безпеки обміну даними, що ставить нові завдання перед архітектурою мереж і протоколами передавання даних.

В архітектурі IoT особливу роль відіграють протоколи рівня застосунків, які визначають логіку взаємодії між елементами системи, моделі обміну повідомленнями, формати даних, механізми підтвердження доставки, узгодження якості обслуговування та інтеграцію з прикладними компонентами.

На відміну від класичного Інтернету, де переважають універсальні протоколи прикладного рівня, у середовищі IoT набули широкого поширення спеціалізовані протоколи, орієнтовані на ресурсно-обмежені пристрої та нестабільні мережеві середовища. До них належать, зокрема, MQTT, DDS, CoAP, AMQP, різноманітні реалізації HTTP, а також протоколи, вбудовані в платформи промислового Інтернету речей.

Особливістю IoT-систем є поєднання великої кількості різноманітних пристроїв, а також різних технологій доступу на фізичному та канальному рівнях (Wi-Fi, стільникові мережі, LPWAN тощо), а також різноманітних вимог до характеристик обміну даними залежно від прикладної області. Для моніторингу довкілля критичною може бути енергоефективність та довготривала автономна робота сенсорів; для промислових застосунків – гарантована доставка повідомлень і передбачувані затримки, для систем

розумної енергетики – масштабованість і стійкість до відмов окремих вузлів. Це приводить до того, що універсального «найкращого» протоколу рівня застосунків для всіх IoT-сценаріїв не існує, а вибір рішення повинен ґрунтуватися на комплексному порівняльному аналізі.

Наявні дослідження та певні галузеві рекомендації [27] здебільшого зосереджуються на описі окремих протоколів (наприклад, MQTT чи CoAP) або узагальненні їхніх властивостей без достатньо глибокого врахування конкретних архітектурних обмежень і моделей трафіку. Варто зазначити, що часто аналіз зводиться до теоретичного порівняння наборів функцій, не підкріпленого експериментальними вимірюваннями в типових для IoT умовах: обмеження смуги пропускання та енергоспоживання, обмежений обсяг буферної пам’яті, змінна топологія мережі тощо. Унаслідок цього розробники та проєктувальники IoT-рішень змушені орієнтуватися на фрагментарні відомості, що не завжди дозволяє обґрунтовано оцінити компроміси між надійністю, затримками, експлуатаційними витратами та складністю реалізації.

Актуальність роботи полягає в розробленні підходів до комплексної оцінки ефективності протоколів рівня застосунків в архітектурі Інтернету речей, що поєднують теоретичний аналіз їхніх властивостей із експериментальними дослідженнями у модельних сценаріях, наближених до реальних умов функціонування. Результати такого дослідження мають забезпечити можливість обґрунтованого вибору протоколу та його параметрів під конкретний клас задач і обмеження, що, у свою чергу, підвищить ефективність, надійність та безпеку IoT-систем.

Об’єктом дослідження у магістерській дисертації є процес обміну даними між компонентами розподілених систем Інтернету речей.

Предметом дослідження є протоколи рівня застосунків в архітектурі IoT, їх функціональні можливості, експлуатаційні характеристики та особливості використання в різних типових сценаріях побудови IoT-систем.

Наукова новизна полягає в тому, що було запропоновано підхід до комплексної оцінки ефективності протоколів рівня застосунків в архітектурі Інтернету речей, який поєднує аналітичне моделювання характеристик протоколів із експериментальними дослідженнями в умовах, наближених до реальних сценаріїв функціонування IoT-систем. Також узагальнено систему критеріїв порівняння протоколів прикладного рівня для IoT (затримка, пропускна здатність, надійність, експлуатаційні витрати, стійкість до втрат, чутливість до параметрів мережі), що враховує специфіку ресурсно-обмежених пристроїв і гетерогенних мережевих середовищ.

Практичне значення роботи полягає в тому, що результати дослідження можуть бути використані при проєктуванні та модернізації систем Інтернету речей для промислових, енергетичних, транспортних, міських та інших застосувань. Розроблені рекомендації щодо вибору протоколів рівня застосунків та їх конфігурації дають змогу зменшити експлуатаційні витрати на передачу даних, підвищити надійність і стійкість систем до збоїв мережевої інфраструктури, а також спростити інтеграцію IoT-рішень із наявними інформаційними системами.

1 ТЕОРЕТИЧНІ ОСНОВИ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Сутність та базові принципи функціонування Інтернету речей

Інтернет речей (Internet of Things, IoT) був визначений у Рекомендації ITU-T Y.2060 (06/2012) як глобальна інфраструктура інформаційного суспільства, що дозволяє надавати передові послуги шляхом взаємозв'язку (фізичних та віртуальних) речей на основі існуючих та сумісних інформаційно-комунікаційних технологій, що розвиваються [1,2].

Базові принципи функціонування IoT безпосередньо пов’язані з глобальною підключеністю, уніфікованою адресацією та орієнтацією на дані, що відповідає концепції «any time (у будь-який час), any place (у будь-якому місці), any thing communication (будь-яка комунікація)», закладеній у Y.4000 [2].

IoT передбачає можливість підключення пристроїв «будь-де та будь-коли» через використання як традиційних мережевих технологій (IP-мережі загального користування), так і спеціалізованих бездротових рішень для пристроїв з обмеженими ресурсами (LPWAN, IEEE 802.15.4, Bluetooth Low Energy тощо) [3,4].

Уніфікована ідентифікація та адресація «речей» (ідентифікатори, адреси, мітки) є передумовою масштабованості IoT-систем, оскільки забезпечує однозначне встановлення зв’язку між об’єктом і сервісами IoT в глобальній інфраструктурі [3,4].

Інтернет речей будується за принципом управління на основі даних: пристрої безперервно збирають телеметричні дані, передають їх через шлюзи на периферійні чи хмарні платформи, де відбуваються зберігання, аналітика й формування керуючих впливів, що узгоджується з референтною моделлю обробки даних в ITU-T [3,4].

З позиції кінцевого користувача IoT проявляється у різноманітних сервісах, які забезпечують автоматизацію побутових, виробничих та міських процесів. Пропонується декілька підходів до класифікації Інтернету речей,

але найбільш поширеним є поділ за типом користувачів та призначенням систем. Такий підхід дозволяє виділити споживчий, комерційний, промисловий та інфраструктурний Інтернет речей, кожен з яких має власні особливості щодо вимог до надійності, безпеки й масштабованості [5].

Споживчий Інтернет речей (Consumer IoT) охоплює пристрої й сервіси, орієнтовані насамперед на кінцевого користувача у побуті. Сюди входять системи «розумного дому», портативні гаджети, «розумна» побутова техніка, домашні системи безпеки та мультимедійні комплекси, які можуть керуватися зі смартфона або через голосових асистентів (рисунок 1.1). Головними характеристиками таких рішень є зручність у використанні, простота інсталяції, а також орієнтація на підвищення комфорту, економію ресурсів та базову безпеку житла. При цьому вимоги до безвідмовності та інформаційної безпеки зазвичай нижчі, ніж у промислових або критичних системах, що відображається на обраних архітектурних рішеннях та політиках захисту [5].

Рисунок 1.1 – Технології системи «Розумний дім» [29]

Комерційний Інтернет речей (Commercial IoT) пов’язаний із використанням IoT-технологій у комерційних організаціях та сервісних компаніях поза сферою важкої промисловості. До цієї групи належать рішення для роздрібної торгівлі, логістики, офісних комплексів, медичних закладів тощо. У роздрібній торгівлі прикладами можуть бути «розумні» полиці з датчиками наявності товару, RFID-мітки для відстеження руху продукції (рисунок 1.2), системи аналізу поведінки відвідувачів на основі даних з відеокамер та сенсорів присутності. У логістиці та складських комплексах застосовуються системи відстеження місцеположення вантажів у реальному часі, моніторинг температури та вологості для чутливих товарів, а також інтелектуальні рішення для оптимізації складських операцій. Для комерційного IoT важливою є інтеграція з існуючими інформаційними системами підприємств, а також можливість масштабування рішень відповідно до зростання бізнесу [5].

Рисунок 1.2 – Процес передачі даних за допомогою RFID-мітки [30]

Промисловий Інтернет речей (Industrial IoT, IIoT) (рисунок 1.3) є одним з найбільш технологічно складних різновидів IoT. У цьому випадку IoT-технології застосовуються у виробничих процесах, енергетиці, на транспорті,

у видобувній промисловості, хімічній та металургійній галузях. Ключову роль відіграють промислові сенсори, контролери, системи автоматизації, що забезпечують безперервний моніторинг параметрів обладнання (температури, вібрацій, тиску, струму), виявлення відхилень та реалізацію концепції прогнозного обслуговування. Застосування IIoT дозволяє зменшити кількість аварій та простоїв, підвищити продуктивність, а також автоматизувати збір технологічних даних для систем підтримки прийняття рішень на різних рівнях управління підприємством [5].

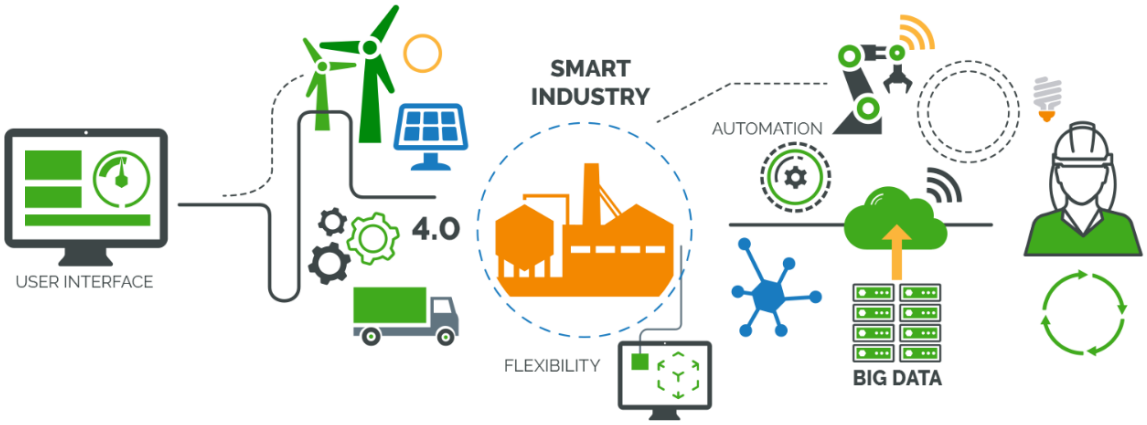


Рисунок 1.3 – Промисловий Інтернет речей [31]

Особливу увагу у промисловому IoT приділяють вимогам до надійності, кібербезпеки та роботи в умовах реального часу. На відміну від споживчих систем, вихід з ладу або компрометація промислової IoT-інфраструктури може призвести до значних економічних збитків, порушення технологічних процесів та загроз для життя і здоров'я персоналу. Тому архітектура IIoT-систем орієнтована на резервування критичних компонентів, використання спеціалізованих промислових протоколів, сегментацію мережі та суворий контроль доступу до обладнання й даних [5].

Інфраструктурний Інтернет речей пов'язаний із розвитком концепції «розумного міста» (Smart City), а також модернізацією критичних інфраструктур. Тут IoT використовується для моніторингу та керування

міським освітленням, дорожнім рухом, громадським транспортом, а також для контролю якості повітря та шумового рівня.

На рисунку 1.4 зображено системи «розумного» вуличного освітлення, що дозволяють автоматично регулювати інтенсивність освітлення залежно від часу доби, наявності пішоходів і транспортних засобів, що сприяє зменшенню споживання електроенергії та підвищенню безпеки на вулицях.

Рисунок 1.4 – Різні варіанти «розумного» освітлення[32]

Інші приклади включають адаптивні світлофори, системи інформування водіїв про завантаженість доріг, розумні парковки з датчиками вільних місць [5] як на рисунку 1.5.

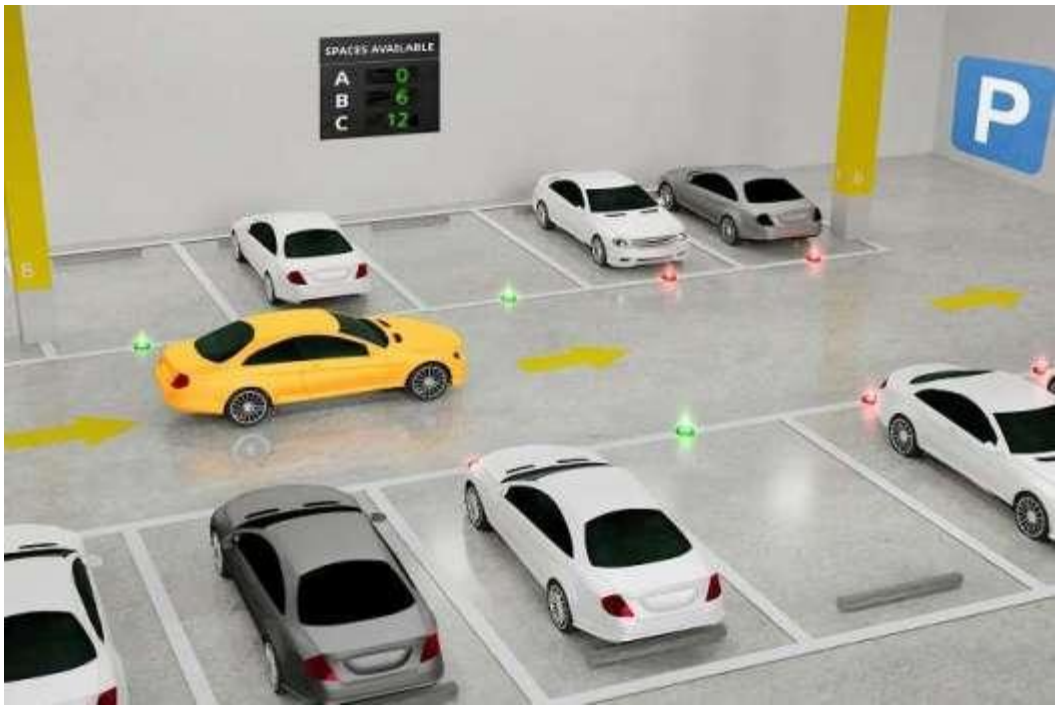


Рисунок 1.5 – Промисловий Інтернет речей [33]

Вимоги до інфраструктурного IoT значною мірою збігаються з вимогами до промислових систем: надвисока надійність, стійкість до збоїв, захищеність каналів зв’язку та вузлів, можливість централізованого моніторингу та управління у масштабі великих територій. Оскільки йдеться про критичні сервіси для населення, будь-які порушення в роботі таких систем можуть мати значний соціальний та економічний ефект, а також викликати загрози громадській безпеці. У зв’язку з цим, проєктування та впровадження інфраструктурного Інтернету речей потребує комплексного підходу, який поєднує технічні рішення, нормативно-правове регулювання та організаційні заходи [5].

Важливо підкреслити, що розглянуті різновиди Інтернету речей не існують ізольовано, а часто перетинаються й взаємодіють між собою. Наприклад, у «розумному» місті поєднуються елементи інфраструктурного IoT (керування освітленням, транспортом, комунальними мережами), промислового IoT (енергетичні й виробничі об’єкти в межах міської інфраструктури) та споживчого IoT (розумні будинки, персональні пристрої

людей). У сфері охорони здоров’я спостерігається інтеграція персональних пристроїв з клінічними системами моніторингу пацієнтів та інформаційними системами лікарень, що наближає такі рішення до комерційного та інфраструктурного сегментів [5].

1.2 Архітектурні моделі IoT

Архітектура Інтернету речей описує логічну структуру системи, яка зв’язує фізичні об’єкти, мережеву інфраструктуру, програмні платформи та бізнес-процеси в єдину інтегровану екосистему. Під час досліджень виділяють кілька типових багаторівневих моделей, що відрізняються ступенем деталізації: трирівневу, чотирирівневу, п’ятирівневу та семирівневу. Кожна з них по-своєму описує шлях даних: від моменту їхнього виникнення на сенсорах до використання в прикладних сервісах і управлінських рішеннях [6].

1.2.1 Трирівнева архітектура IoT

На рисунку 1.6 зображена трирівнева архітектура, що є досить поширеною та загальновідомою. Вперше була використана на початкових етапах дослідження Інтернету речей. Складається з трьох рівнів: сенсорів (perception), мережевого (network) і прикладного (application). Вона є базовою логічною моделлю, що описує шлях даних від фізичних пристроїв до прикладних сервісів [6, 7].

Трирівнева архітектура використовується як спрощена, але формальна схема, на основі якої будуються більш деталізовані 4-, 5- та 7-рівневі моделі. Її перевага – простота: кожен рівень має чітку роль (збір даних, передавання, використання), а взаємодія між рівнями описується через стандартизовані інтерфейси та протоколи [6].

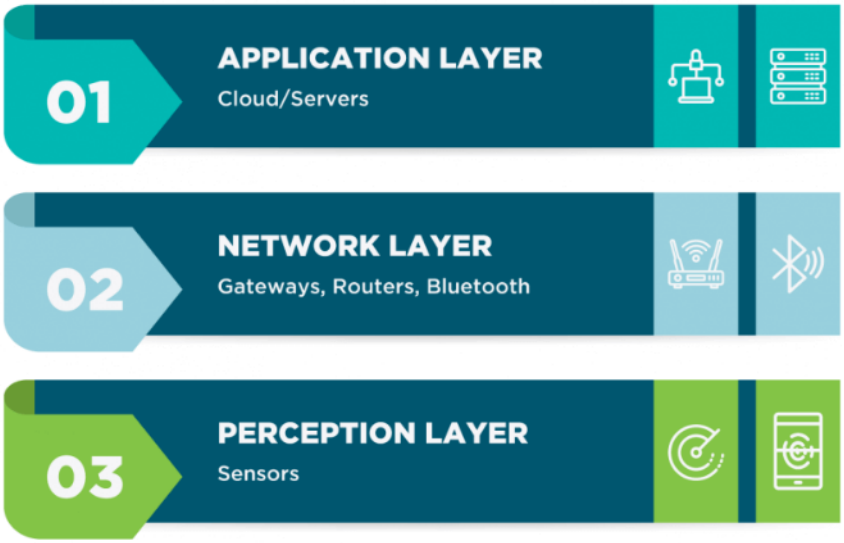


Рисунок 1.6 – Трирівнева архітектура мережі IoT [6]

Найнижчий рівень в даній архітектурі - це Рівень сприйняття (Perception Layer). Він відповідає за взаємодію з фізичним світом. У ньому в основному використовуються датчики та вбудовані системи. Вони збирають великі обсяги даних на основі вимог. Це також включає периферійні пристрої, датчики та виконавчі механізми, які взаємодіють з навколишнім середовищем. Вони виявляють певні просторові параметри або виявляють інші інтелектуальні речі чи об'єкти в навколишньому середовищі [6, 7, 8].

До типових компонентів відносяться:

1. Сенсори: аналогові й цифрові датчики (температури, вологості, газів, струму або напруги, прискорення, положення, освітлення, звуку), що формують виміряні значення з певною періодичністю або за подіями.
2. Пристрої ідентифікації: RFID-мітки та зчитувачі, NFC, штрих-коди, QR-коди, що забезпечують унікальну ідентифікацію об’єктів у системі.
3. Актуатори: клапани, електроприводи, двигуни, світлові та звукові індикатори, що реалізують керуючі впливи на фізичні об’єкти (включення/вимикання, зміна режиму роботи обладнання тощо).

4. Вбудовані пристрої/контролери: мікроконтролери, одноплатні комп’ютери, промислові контролери (PLC), які збирають дані від кількох сенсорів/актуаторів і забезпечують базову локальну логіку [6].

Рівень сприйняття відповідає за збір даних: перетворення фізичних величин у цифрові показники із заданою частотою, роздільною здатністю й точністю. Також тут відбувається попередня обробка: фільтрація шумів, форматування даних у структуру, придатну до передавання (кадри, пакети, повідомлення). І на останок локальне керування: виконання простих алгоритмів реагування без участі хмари (наприклад, аварійне відключення при перевищенні критичного значення, підтримання температури в межах діапазону) [6].

Щодо обмеженості, то більшість сенсорних вузлів має низьке енергоспоживання, малий обсяг пам’яті й обчислювальних ресурсів, що обмежує складність алгоритмів на цьому рівні.

Варто зазначити, що пристрої можуть працювати у складних умовах (температура, вібрація, пил, волога), часто – з ризиком фізичного доступу зловмисників, тому важливі механічна міцність, захист корпусу та базові засоби захисту прошивки [6, 8].

Мережевий рівень (Network Layer), що забезпечує транспортування даних, до обчислювальних ресурсів (сервера, шлюзів, хмарних платформ) та доставку керуючих команд у зворотному напрямку [6, 7].

До типових технології відносяться:

- Бездротові технології ближнього радіуса: Wi-Fi, Bluetooth/BLE, ZigBee, 6LoWPAN – використовуються для локальних сенсорних мереж і домашніх/офісних рішень.
- Технології великого радіуса (LPWAN): LoRaWAN, Sigfox, NB-IoT, LTE-M – застосовуються для широкомасштабних розгортань із низьким енергоспоживанням.

- Стільникові мережі: 3G/4G/5G – для мобільних-розподілених систем з потребою у глобальному покритті.
- Дротові технології: Ethernet, оптоволокно, PLC, промислові шини (Modbus, CAN), які використовуються в промисловості та інфраструктурі з підвищеними вимогами до надійності.
- Мережеві вузли: маршрутизатори, комутатори, шлюзи, концентратори (hubs), які з'єднують сенсорні підмережі з IP-мережею та/або хмарою [6].

Основними функціями мережевого рівня являється:

- Адресація й маршрутизація: призначення адрес пристроям (часто на основі IPv6, 6LoWPAN) та вибір маршрутів для доставки пакета до цільового вузла.
- Передавання даних: інкапсуляція, фрагментація, повторна передача при помилках, управління чергами, підтримка різних класів сервісу (телеметрія, команди керування, оновлення ПЗ).
- Протокольна конвертація: перетворення специфічних протоколів сенсорних мереж у стандартний IP-стек / протоколи верхніх рівнів.
- Безпека передавання: використання криптографічних протоколів (TLS/DTLS, IPsec, VPN-тунелі), аутентифікація вузлів і базовий контроль доступу на рівні мережі [6, 7, 8].

Даний рівень також має свої особливості та вимоги: потребує масштабованості, так як мережева інфраструктура повинна підтримувати підключення великої кількості однотипних пристроїв, часто з обмеженою смугою пропускання. Важливим критерієм являється якість обслуговування, так як різні типи трафіку можуть мати різні вимоги до затримки, надійності й пропускної здатності. Також не слід забувати і про енергоефективність, бо для батарейних вузлів критичною є підтримка протоколів з низьким енергоспоживанням [6, 7, 8].

Прикладний рівень (Application Layer) є верхнім рівнем трирівневої архітектури, де дані перетворюються на конкретні сервіси, доступні користувачам, операторам і бізнес-системам. Наприклад: у застосунку для розумного дому, де користувачі натискають кнопку в застосунку, щоб увімкнути кавоварку. Прикладний рівень відповідає за надання клієнту ресурсів, специфічних для застосунку. Він визначає різні способи використання Інтернету речей, такі як розумні будинки, розумні міста та розумне здоров'я [6, 7].

Основними функціями рівня застосунків являються: інтерпретація й візуалізація даних (побудова графіків, панелей моніторингу, звітів, генерація сповіщень). Також реалізація бізнес-логіки: різні правила автоматизації, сценарії керування, створення профілів користувачів. І на останок керування пристроями: відправка команд, зміна конфігурації, оновлення прошивки [6, 7].

Щодо особливостей даного рівню, то важливими критеріями є масштабованість сервісів, адже застосунки мають обробляти значні обсяги даних у реальному часі, підтримувати велику кількість одночасних користувачів та пристроїв. Не менш важливою є гнучкість: можливість додавати нові функції без зміни нижніх рівнів архітектури, використовуючи стандартизовані API. Щодо безпеки та конфіденційності, то важливим є контроль доступу до даних, аутентифікація користувачів, аудит операцій, дотримання нормативних вимог щодо персональних даних [6, 7].

1.2.2 Чотирирівнева архітектура IoT

На рисунку 1.7 зображена архітектура, яка використовує чотири рівні Інтернету речей, вона схожа на базову. Дані також збираються датчиками та передаються до системи зберігання мережевим рівнем. Однак ця еталонна архітектура має ще один рівень, відомий як рівень підтримки (support). Він допомагає вирішити основну проблему базової архітектури IoT: слабку

безпеку. Рівень підтримки розташований між рівнем мережі та прикладним рівнем (application) [6, 9].

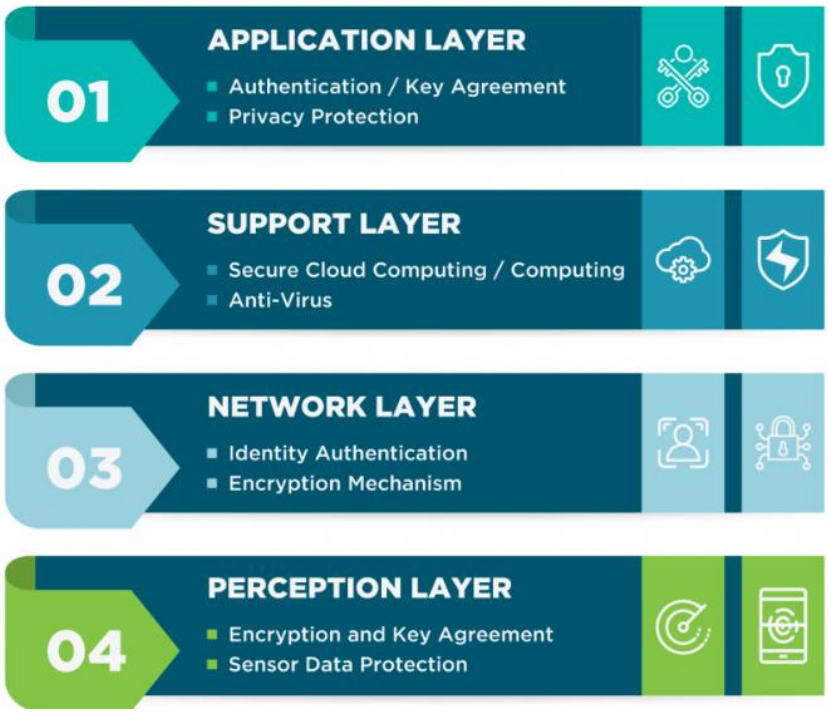


Рисунок 1.7 – Чотирирівнева архітектура мережі IoT [6]

Трирівнева структура Інтернету речей стикалася з багатьма загрозами. Тому було запропоновано додати ще один рівень, який міг би вирішити більшість проблем. Він також відомий як «Безпечний» (Secure), оскільки зосереджений на тому, щоб зробити моделі комунікації IoT більш захищеними від можливих загроз [6].

Рівень підтримки перевіряє, чи інформація надіслана від автентифікованого користувача. У більшості випадків як методи автентифікації використовуються паролі або спільні ключі. Після перевірки користувача рівень підтримки надсилає інформацію, отриману від датчиків пристроїв Інтернету речей, на мережевий рівень. Це третій рівень у цій багаторівневій архітектурі Інтернету речей [6].

Хоч і рівень підтримки допомагає підвищити безпеку, однак він не захищає всі рівні архітектури IoT від усіх інших загроз. Дві найпоширеніші загрози такі:

- DoS-атака (відмова в обслуговуванні). Хакер повинен надсилати багато запитів на мережевий рівень, тому система не зможе обробити їх усі одночасно. У такому випадку перевантажена система перестане працювати.
- Внутрішня атака. Для її проведення шахраю необхідно отримати облікові дані для входу існуючого користувача системи Інтернету речей. Після цього він може завантажити в систему шкідливий код , щоб зруйнувати її або отримати конфіденційну інформацію [6, 9].

1.2.3 П’ятирівнева архітектура IoT

Модель, що передбачає чотири рівні архітектури Інтернету речей, була гарним рішенням. Вона допомогла підвищити безпеку логічного проєкту Інтернету речей. Однак не була ідеальною, адже не відповідала всім вимогам користувачів, також існували деякі проблеми безпеки. Тому була розроблена п'ятирівнева архітектура, яка зображена на рисунку 1.8. Вона складається з базової архітектури та передбачає два додаткові рівні: обробки (processing) та бізнес (business) [6, 10].

Рисунок 1.8 – П’ятирівнева архітектура мережі IoT [6]

Рівень processing є центральним елементом п’ятирівневої архітектури й відповідає за уніфіковану обробку, зберігання та надання даних і сервісів. На цьому рівні зазвичай розміщується IoT-платформа, яка включає брокери повідомлень (наприклад, MQTT чи AMQP), служби збору та зберігання (NoSQL-сховища), сервіси аналітики та керування пристроями [6, 10].

До основних функції належать: прийом і нормалізація даних з різних транспортних каналів, очищення та агрегація, реалізація правил і сценаріїв автоматизації, централізоване управління життєвим циклом пристроїв і надання стандартизованих інтерфейсів верхнім рівням [6, 10].

Бізнес-рівень інтерпретує результати роботи IoT-системи у термінах бізнес-показників, процесів і стратегій. На цьому рівні формуються та відстежуються ключові показники ефективності (KPI), визначаються бізнес-моделі та політики (наприклад, тарифи, моделі монетизації «IoT-як-сервіс»), виконується інтеграція з ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), BPM (Business Process Management) та системами бізнес-аналітики (BI) [6, 10].

Таким чином п’ятирівнева архітектура чітко пов’язує технічну інфраструктуру IoT з цілями й процесами організації, що робить її зручною для опису корпоративних та галузевих IoT-рішень [6, 10].

1.2.4 Семирівнева архітектура IoT

Семирівнева архітектура IoT (IoT World Forum / Cisco IoT Reference Model) (рисунок 1.9) описує IoT-систему як послідовність із семи рівнів, через які дані проходять шлях від фізичних пристроїв до бізнес-процесів. Такий підхід дозволяє детально рознести функції збору, передавання, зберігання, підготовки, використання та організаційної інтеграції даних [6].

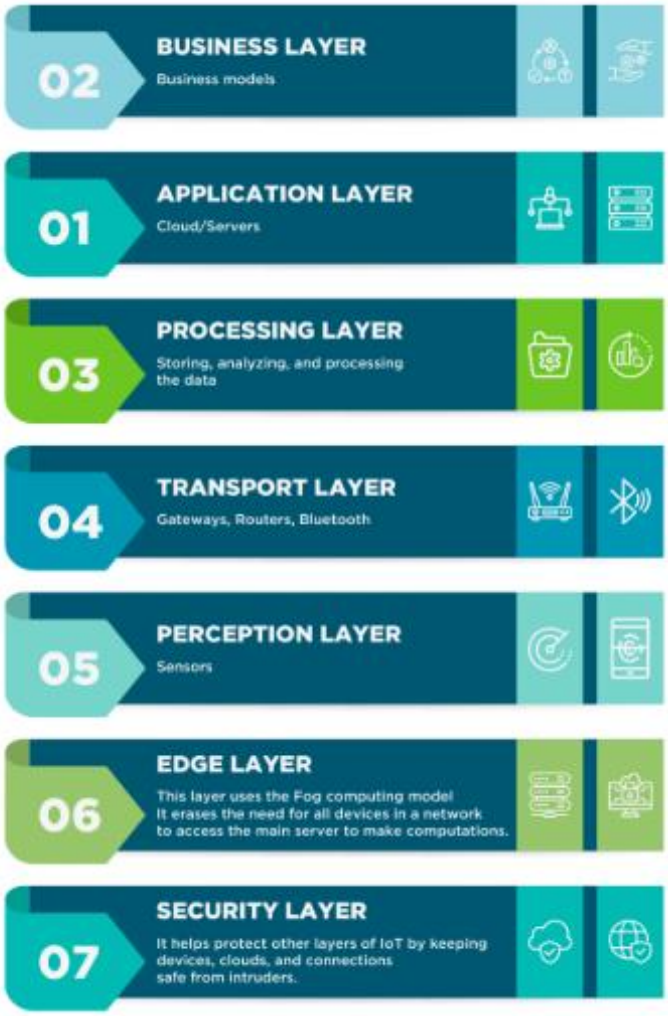


Рисунок 1.9 – Семирівнева архітектура мережі IoT [6]

Edge Layer (периферійний рівень) є додатковим рівнем, який розширює класичну архітектуру IoT. Необхідність його появи зумовлена стрімким зростанням обсягів даних, що генеруються IoT-пристроями, та потребою зменшити затримку при їх обробленні. Цей рівень розміщується між мережевим і обчислювальним рівнями. Він реалізує принцип fog computing (туманне обчислення), завдяки чому частина обчислень і аналізу даних виконується безпосередньо поблизу джерела даних. Це дозволяє знизити навантаження на центральну хмарну інфраструктуру, підвищити продуктивність системи, а також забезпечити оперативне реагування в реальному часі. Таким чином, Edge Layer виступає проміжним рівнем інтелекту, що сприяє побудові масштабованих і надійних IoT-систем [6].

Рівень безпеки (Security Layer) є критично важливою складовою архітектури IoT, оскільки кількість потенційних загроз зростає пропорційно до кількості з’єднаних пристроїв. Він охоплює широкий спектр механізмів — від аутентифікації та шифрування даних до контролю доступу, моніторингу аномальної активності та реагування на інциденти. У сучасних архітектурах безпека розглядається не як окремий рівень, а як наскрізна функція, яка інтегрується на всіх етапах життєвого циклу IoT-системи — від пристроїв до хмарних сервісів. Такий підхід забезпечує цілісність, конфіденційність і доступність даних в умовах динамічного середовища IoT [6].

1.2.5 Порівняльний аналіз архітектурних моделей IoT

Вибір конкретної архітектури безпосередньо впливає на складність проєктування, вимоги до апаратних і програмних ресурсів, можливості масштабування, інтеграції з хмарними платформами та системами бізнес-аналітики, а також на спосіб організації механізмів інформаційної безпеки. З огляду на це доцільно виконати порівняльний аналіз трирівневої, чотирирівневої, п’ятирівневої та семирівневої моделей за ключовими критеріями, що наведені у таблиці 1.1.

Таблиця 1.1 – Порівняння архітектурних моделей IoT за ключовими критеріями [6, 7, 8, 9, 10]

Критерій	3-рівнева модель	4-рівнева модель	5-рівнева модель	7-рівнева модель
Призначення	Збір, передавання, використання даних	Чітке виокремлення рівню обробки між мережевим рівнем і рівнем застосунків.	Додатковий бізнес-рівень для інтеграції з управлінськими рішеннями	Деталізований поділ, чіткий розподіл функцій від пристроїв до бізнес-процесів
Складність реалізації	Низька	Середня	Вища	Дуже висока
Масштабованість	Обмежена: важко розносити навантаження між проміжними рівнями	Задовільна: частину обробки можна винести на окремий рівень (хмара)	Висока: окремі рівні обробки й бізнес-логіки спрощують масштабування за функціями	Максимальна: окремі рівні накопичення, абстракції, застосунків і процесів дозволяють незалежно масштабувати кожен домен
Придатність до критичних застосувань	Обмежена; підходить здебільшого для малих/локальних рішень (smart home)	Придатна для середніх рішень (розподілений моніторинг, базова автоматизація)	Добра підтримка промислових і міських застосувань завдяки бізнес-рівню та middleware	Орієнтована на великі промислові й корпоративні екосистеми (Industrial IoT, smart city)
Модель посилань / стандарти	Узагальнені академічні описи трирівневої архітектури	Узгоджується з IoT-reference-моделлю ITU-T Y.2060	Описана в низці оглядових робіт як 5-layer IoT	Відповідає Cisco IoT Reference Model
Можливості безпеки	Базова, часто реалізується без детального рознесення контролів	Легше впровадити захист на транспортному та обробному рівнях	Дозволяє будувати наскрізну політику безпеки	Підтримує багаторівневу безпеку з окремими механізмами

1.3 Апаратні та програмні компоненти архітектур Інтернету речей

Апаратні та програмні компоненти доцільно класифікувати за роллю в багаторівневій архітектурі: рівень пристроїв, мережевий, рівень обробки й

платформи, рівень застосунків та бізнес-систем. Такий поділ показує повний технологічний ланцюг від фізичного середовища до бізнес-рішень [11, 12].

Що до апаратних компонентів, то на рівні сприйняття - це все обладнання, яке безпосередньо контактує з фізичним світом.

- Сенсорні модулі:
    - Фізичні датчики: температури, вологості, барометричні сенсори, сенсори освітленості, ультразвукові далекоміри, газові сенсори, струму/напруги, лічильні сенсори (витрати, обертів), біомедичні сенсори (пульс, тиск).
    - Модульні сенсорні плати: комбіновані модулі «температура + вологість + тиск».
- Пристрої ідентифікації:
    - Пасивні або активні RFID-мітки та стаціонарні/мобільні RFID-зчитувачі.
    - NFC-антени й чипи, сканери штрих- та QR-кодів.
- Контролери та MCU-плати:
    - Мікроконтролери загального призначення (STM32 (рис. 1.10), AVR, ESP32), що інтегрують цифрові інтерфейси, таймери.

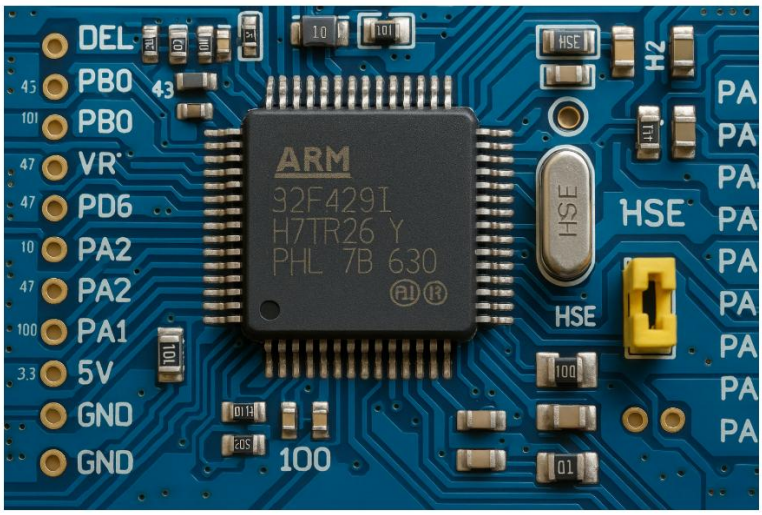


Рисунок 1.10 – STM32 [28]

- Одноплатні комп’ютери (Raspberry Pi (рисунок 1.11), BeagleBone (рисунок 1.12)) для більш складної локальної обробки (штучний інтелект, відео-аналітика).

Рисунок 1.11 – Raspberry Pi [34]

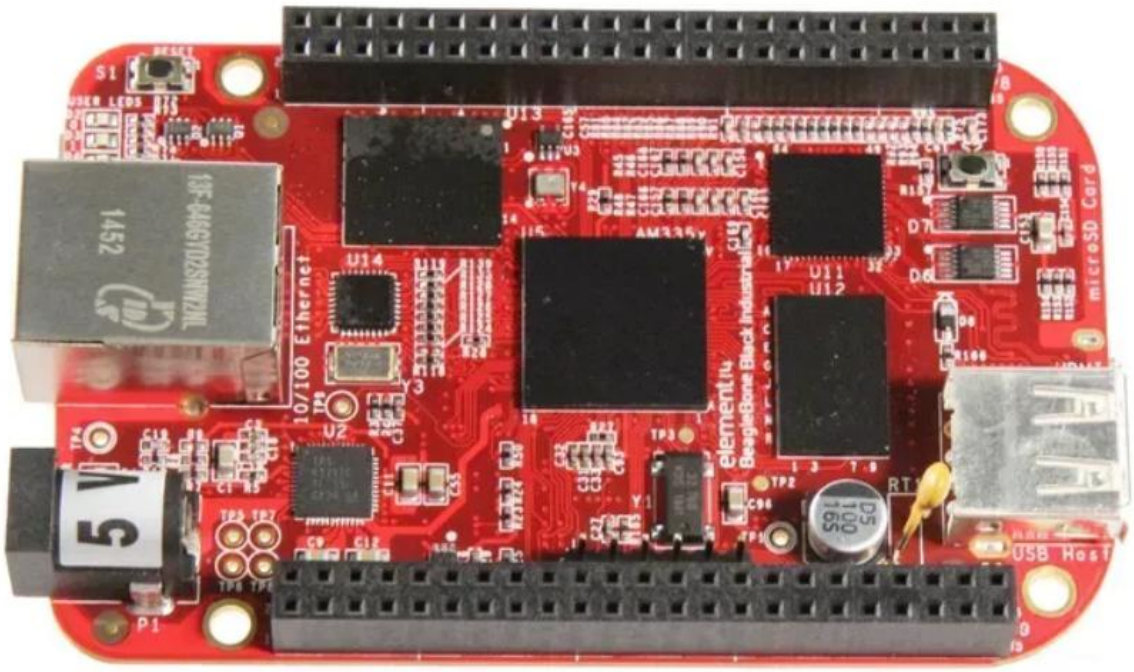


Рисунок 1.12 –Мікрокомп'ютер BeagleBone Black Industrial [35]

Ці пристрої визначають якість, частоту й надійність первинних даних, а також здатність системи впливати на об’єкт керування [11, 12].

Мережеві апаратні компоненти відповідають за фізичну й канальну частину підключення.

- Радіоінтерфейси:
    - Короткого радіуса дії: Wi-Fi, Bluetooth/BLE, ZigBee, 6LoWPAN – для локальних мереж у будівлях, цехах, кампусах.
    - Далекого радіуса / LPWAN: LoRa/LoRaWAN, Sigfox, NB-IoT, LTE-M – для масових розгортань із малим енергоспоживанням та великим радіусом дії (смарт-лічильники, агромоніторинг).
- IoT-шлюзи та конвертори:
    - Агрегують дані з протоколів (Modbus RTU/TCP, CAN, OPC-UA, власні радіопротоколи) й транслюють їх у IP-з’єднання до платформ (через Ethernet, стільниковий зв’язок).
    - Можуть мати апаратне прискорення криптографії, резервні канали зв’язку, вбудовані механізми безпеки.
- Комутатори та маршрутизатори:
    - Комутатори, що працюють на канальному або мережевому рівнях моделі OSI, створення VLAN, приоритизації трафіку.
    - Маршрутизатори з підтримкою VPN, QoS, фаєрволів.
- Базові станції:
    - LPWAN-шлюзи (LoRaWAN gateways, eNodeB для NB-IoT/LTE-M), що обслуговують великі зони покриття й багато вузлів.

Ці компоненти визначають топологію, покриття, пропускну здатність та латентність системи, а також межі сегментації з точки зору безпеки. [11, 12].

Апаратні ресурси обробки та зберігання .

- Edge-сервери та промислові ПК:

    - Виконують попередню аналітику, фільтрацію, кешування й локальне рішення задач з підвищеними вимогами до затримки (наприклад, аварійне реагування).
    - Часто мають захищене виконання (підвищена стійкість, широкі температурні діапазони).
- Сервери ЦОД:
    - Утримують кластери БД, брокерів, аналітичних та інтеграційних сервісів.
    - Реалізуються як фізичні сервери, VM або контейнерні кластери в публічній чи приватній хмарі.

Від їхньої конфігурації залежать масштабування, відмовостійкість і загальна продуктивність платформи [11, 12].

Апаратні елементи верхніх рівнів.

- Робочі місця й інтерфейси:
    - ПК, ноутбуки, HMI-панелі, операторські консолі, планшети й смартфони.
- Сервери бізнес-систем:
    - Сервери ERP/CRM/BI/BPM, що можуть бути окремими або частиною загальної хмарної інфраструктури [11, 12].

Програмне забезпечення, яке виконується безпосередньо на сенсорах, контролерах і edge-пристроях.

- Firmware (прошивка):
    - Основна логіка опитування сенсорів, реалізація простих автоматичних реакцій.
    - Зазвичай включає енергозберігаючі режими, обробку помилок і базову діагностику.
- Драйвери та HAL (hardware abstraction layer):
    - Інкапсулюють роботу з периферією, дозволяють переносити код між різними апаратними платформами.

- Прикладні протоколи:
    - MQTT (publish/subscribe), CoAP (REST-подібний over UDP), HTTP/HTTPS, AMQP– реалізуються як бібліотеки на пристроях і як серверні компоненти на брокерах/шлюзах.
- ПЗ мережевих компонентів:
    - ОС маршрутизаторів/шлюзів (включно з функціями NAT, VPN, firewall), системи IDS/IPS, що аналізують трафік на наявність аномалій та атак.

Цей рівень визначає схеми взаємодії, інтероперабельність і базову безпеку передачі даних [11, 12].

Проміжне програмне забезпечення та IoT-платформи. Ключовий елемент зв’язування пристроїв і застосунків.

- Комунікаційні сервіси:
    - MQTT/AMQP-брокери, HTTP/CoAP-сервери, що приймають, маршрутизують повідомлення, реалізують QoS-рівні, зберігають тимчасові черги.
- Сховища даних:
    - Time-series БД для телеметрії (InfluxDB, TimescaleDB), NoSQL для напівструктурованих даних, об’єктні сховища для даних журналу подій/log-даних.
- Device management:
    - Реєстр пристроїв (ідентифікатори, сертифікати, метадані), управління конфігурацією, OTA-оновлення, віддалена діагностика й відключення.

Проміжне програмне забезпечення ізолює прикладних розробників від складності низьких рівнів і дозволяє сприймати IoT як набір сервісів: «дані», «події», «керування пристроями».

Прикладні та бізнес-орієнтовані програмні компоненти. Верхній рівень програмного стеку.

- Прикладні IoT-додатки:
    - Веб-панелі для операторів, мобільні додатки для кінцевих користувачів, SCADA-клієнти, -інтерфейси.
    - Реалізують моніторинг, керування, налаштування порогів, конфігурацію сценаріїв.
- Бізнес-системи та інтеграція:
    - ERP (управління ресурсами й запасами), CRM (управління взаємодією з клієнтами), BPM (моделювання й виконання бізнес-процесів), CMMS (управління обслуговуванням обладнання).
    - Конектори та шини даних, які інтегрують IoT-платформи з цими системами [11, 12].

1.4 Загрози та механізми забезпечення безпеки в архітектурах IoT

Загрози в архітектурах IoT доцільно розглядати по рівнях, оскільки атаки спрямовані як на порушення доступності, цілісності та конфіденційності даних, так і на фізичну та функціональну безпеку кіберфізичних систем. Для протидії використовують вбудовану багаторівневу систему безпеки, що поєднує легку криптографію, автентифікацію й авторизацію, сегментацію мережі, моніторинг аномалій, безпечну розробку програмного забезпечення та керування життєвим циклом пристроїв [13].

На рівні пристроїв (perception / device layer) поширені атаки на сенсори: клонування й підміна пристроїв, фальсифікація показників, фізичний саботаж, side-channel атаки, завантаження неавторизованої прошивки, використання вузла як «точки опори» для подальшого проникнення в мережу. Ці атаки реалізуються через слабку або відсутню автентифікацію, відкриті debug-інтерфейси, нестачу механізмів захисту пам’яті й безпечного

завантаження, жорстко прошиті паролі та відсутність захищеного оновлення ПЗ [13].

На мережевому рівні характерними є перехоплення й модифікація трафіку (MitM), підміна вузлів, DoS/DDoS проти шлюзів або хмарних сервісів, порт-сканування та експлуатація вразливих стеків протоколів (наприклад, ZigBee чи промислові протоколи OT). Додатковий ризик створює конвергенція IT та OT, коли компрометація звичайної IT-мережі дає зловмиснику доступ до критичних систем керування виробництвом через слабко захищені IoT-вузли [13].

На рівні сервісів і застосунків виникають загрози компрометації API, підробки команд керування, ескалації привілеїв, атак через уразливі веб-інтерфейси та масових витоків телеметричних даних. Для IoT особливо критичною є функціональна безпека: маніпуляція віртуальними даними (V) може спричинити некоректну дію пристрою (T) і, як наслідок, реальну фізичну шкоду об’єктам або людям. [13].

На рисунку 1.13 узагальнено ключові вимоги до безпеки, які мають виконуватися на всіх рівнях архітектури IoT, зокрема ідентифікація користувачів, керування ідентичностями, захищений обмін даними, безпечна мережа, безпечне зберігання, безпечне виконання програм, захищений вміст і стійкість до несанкціонованого доступу. Сумарно вони утворюють вбудовану систему безпеки, що збирає дані з датчиків, аналізує контекст і забезпечує превентивні дії для захисту інформації користувачів та запобігання атакам.

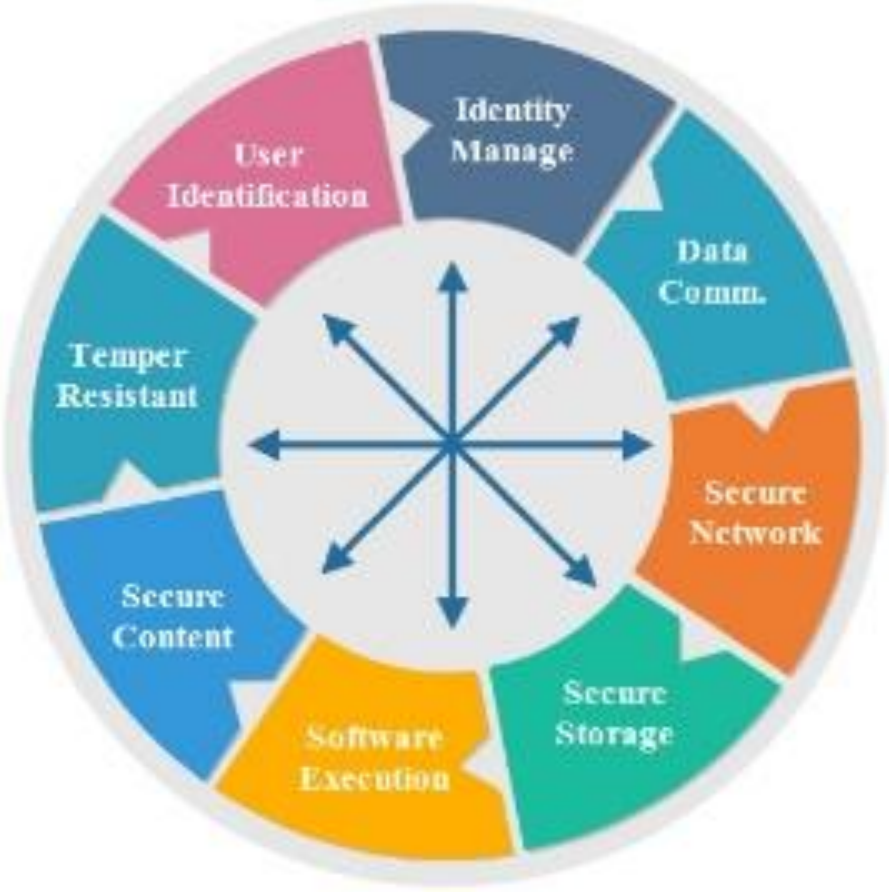


Рисунок 1.13 – Основні проблеми безпеки в IoT [13]

Ідентифікація користувача (User Identification). Це стосується процесу автентифікації користувачів перед тим, як дозволити їм використовувати ресурси системи. Існує багато способів перевірки користувачів, зокрема за допомогою попередньо наданого ключа або пароля [13].

Керування ідентифікацією (Identity Management). Даний процес займається ідентифікацією окремих пристроїв у системі. Він також контролює їхній доступ до ресурсів у цій системі, пов'язуючи права та обмеження користувачів з розпізнаною сутністю [13].

Безпечний обмін даними (Secure Data Communication) вимагає автентифікації пристроїв зв'язку, забезпечення конфіденційності та цілісності переданих даних і захисту об'єктів пристроїв зв'язку [13].

Безпечна мережа (Secure Network). Безпечний доступ до мережі забезпечує мережеве з’єднання або послугу, лише якщо пристрій

авторизований. Неавторизованим пристроям доступ до мережі заборонено [13].

Безпечне зберігання (Secure Storage). Безпечне сховище відповідає за захист інформації користувачів від зловмисників та зовнішнього моніторингу. Тому воно вимагає конфіденційності та надійності конфіденційної інформації, що зберігається в системі [13].

Безпечне середовище виконання програмного забезпечення (Secure Software Execution Environment). Це стосується захищеного середовища виконання з керованим кодом, розробленого для захисту від неочікуваних програм або програмного забезпечення [13].

Безпечний вміст (Secure Contents). Безпека контенту забезпечує безпеку інформації в системі та захищає її від зловмисників, вірусів та порушників [13].

Захист від несанкціонованого доступу (Tamper Resistance). Існує багато атак, які впливають на систему та викрадають інформацію користувачів без їхнього дозволу. Тому існує потреба в таких механізмах, щоб позбавити зловмисників контролю [13].

Через сукупність наведених атак та вимог формується вбудована система безпеки IoT, яка збирає інформацію з навколишнього середовища за допомогою сенсорів, аналізує потенційні загрози й передає узагальнені дані на центральний вузол або сервер для прийняття захисних рішень. Така система інтегрується в мережеву архітектуру IoT і реалізує функції контролю доступу, фільтрації трафіку, кореляції подій безпеки та автоматичного реагування на інциденти [13].

Ключові функції безпечної архітектури включають використання легкої криптографії для шифрування й аутентифікації з урахуванням обмежених ресурсів IoT-пристроїв, що дозволяє зменшити енергоспоживання та вимоги до пам’яті без втрати базового рівня захисту. Безпечна операційна система забезпечує захищене завантаження, контроль цілісності ПЗ, ізоляцію

процесів і безпечну взаємодію між компонентами, створюючи довірену основу для верхніх рівнів [13].

Фізичний захист охоплює зберігання секретних ключів у захищених апаратних модулях, екранування й антиманіпуляційні корпуси, що ускладнюють фізичний доступ до чутливих даних на пристроях. Безпечне сховище доповнюється механізмами шифрування даних «на диску», керування ключами та політиками резервного копіювання, що знижує ризик витоку інформації навіть у разі компрометації окремих вузлів [13].

Серед важливих властивостей безпечної платформи для IoT можна виділити підтримку безпечних вторинних сховищ, захищене середовище виконання програм, а також безпечний блок керування пам’яттю, який контролює доступ до пам’яті й запобігає експлуатації типових помилок програмування. Сукупність цих механізмів утворює цілісну архітектуру безпеки, яка забезпечує захист даних та сервісів IoT протягом усього життєвого циклу системи, від сенсора до хмарного застосунку [13].

1.5 Висновки з розділу 1

У першому розділі було сформовано цілісне уявлення про те, що таке Інтернет речей і як він побудований на концептуальному рівні. Показано, що IoT доцільно розглядати як багаторівневу інфраструктуру, у якій фізичні та віртуальні об’єкти з можливістю збору, обробки й обміну даними об’єднані в єдине середовище за допомогою мережевих технологій.

У межах розділу було описано, як саме організована структура IoT: від базової моделі «речей», до розподілу функцій між різними рівнями архітектури. Визначальним чинником масштабованості й керованості таких систем є поділ на рівні, які відповідають за збір даних, їх передавання, обробку та використання в прикладних сервісах. Це дозволяє окремо розвивати апаратну частину, мережеву інфраструктуру та програмні рішення, не порушуючи цілісності системи.

Окрему увагу приділено класифікації апаратних і програмних компонентів IoT-екосистеми. Були виділені основні типи пристроїв (датчики, виконавчі механізми, контролери, шлюзи), програмні платформи, сервіси зберігання та аналітики даних, а також прикладні застосунки, що забезпечують взаємодію з користувачем. Показано, як ці елементи поєднуються між собою, утворюючи повноцінну систему, де кожен компонент виконує свою роль у загальному ланцюгу.

Також було проаналізовано основні загрози, характерні для архітектур Інтернету речей, та визначено підходи до забезпечення безпеки. Вразливими є всі рівні – від кінцевих пристроїв до хмарних сервісів, тож потрібні комплексні механізми захисту: автентифікація та авторизація, шифрування даних, безпечне оновлення прошивок, моніторинг подій безпеки. Такий підхід дає змогу ще на теоретичному етапі врахувати аспекти безпеки під час проєктування IoT-рішень.

Отже, теоретичні основи Інтернету речей охоплюють сукупність базових понять, архітектурних підходів, класифікацій компонентів та принципів безпеки, які разом формують методологічну базу для подальших досліджень.

2 ПРОТОКОЛИ РІВНЯ ЗАСТОСУНКІВ В ІОТ: СТРУКТУРА, КЛАСИФІКАЦІЯ ТА АНАЛІЗ

2.1 Роль протоколів рівня застосунків у взаємодії IoT-пристроїв

Протоколи прикладного рівня мають вирішальне значення для забезпечення зв'язку та обміну даними між пристроями Інтернету речей та серверами, або навіть між самими пристроями. Без них пристрої IoT не змогли б обмінюватися інформацією. Вони діють як інтерфейс між пристроєм IoT та мережею, обробляючи форматування та представлення даних [14].

Рівень додатків розташований на найвищому рівні мережевого стеку. Його основна функція в Інтернеті речей полягає в забезпеченні стандартизованого механізму взаємодії між пристроями. Основні завдання цього рівня охоплюють:

- Транспортування даних (Transport Data) відбувається наступним чином: протоколи визначають, як дані упаковуються, адресуються та надсилаються мережею. Це гарантує, що дані надходять правильно та можуть бути інтерпретовані приймаючим пристроєм [14].
- Форматування даних (Format Data): дані мають бути структуровані таким чином, щоб їх розуміли як відправник, так і одержувач. Протоколи прикладного рівня обробляють це форматування, забезпечуючи сумісність [14].
- Представлення даних (Present Data): протокол визначає, як дані представляють програмі, яка їх використовує. Це може включати кодування, шифрування та інші етапи обробки [14].
- Увімкнення взаємодії (Enable Interaction): протоколи визначають, як пристрої ідентифікують та взаємодіють один з одним. Це забезпечує безперебійний зв'язок між різними типами пристроїв Інтернету речей [14].

Таким чином, протоколи рівня застосунків фактично визначають «мову спілкування» IoT-пристроїв: від форматів телеметрії до моделей виклику сервісів, що робить їх вибір критичним для продуктивності, надійності, масштабованості та безпеки всієї системи.

2.2 Класифікація та функціональні особливості основних протоколів прикладного рівня

У системах Інтернету речей застосовується ціла низка протоколів прикладного рівня, однак на практиці домінують декілька, які по-різному балансують між легкістю реалізації, експлуатаційними витратами, надійністю передачі та підтримкою реального часу:

- MQTT (Message Queuing Telemetry Transport);
- DDS (Data Distribution Service);
- CoAP (Constrained Application Protocol);
- AMQP (Advanced Message Queuing Protocol);
- HTTP (Hypertext Transfer Protocol).

2.2.1 MQTT

MQTT — це легкий протокол обміну повідомленнями на основі публікації та підписки, розроблений для пристроїв з обмеженими ресурсами та мереж з низькою пропускною здатністю, високою затримкою або ненадійними мережами [15, 16, 17].

Згідно зі стандартом OASIS: *«MQTT — це протокол передачі повідомлень між клієнт-сервером та клієнтом. Він легкий, відкритий, простий та розроблений таким чином, щоб його було легко впроваджувати»* [17].

Він широко використовується в додатках Інтернету речей, забезпечуючи ефективний зв'язок між датчиками, виконавчими механізмами

та іншими пристроями. Варто зазначити, що MQTT став одним із найкращих протоколів Інтернету речей завдяки своїм унікальним функціям та можливостям, адаптованим до конкретних потреб систем Інтернету речей [15, 16, 17].

IoT-пристрої часто обмежені з точки зору обчислювальної потужності, пам'яті та споживання енергії. Мінімальні експлуатаційні витрати та малий розмір пакетів MQTT роблять його ідеальним для цих пристроїв, оскільки він споживає менше ресурсів, забезпечуючи ефективну комунікацію навіть з обмеженими можливостями. Дослідження показують, що MQTT вимагає до 80% меншої пропускної здатності мережі, ніж HTTP, для передачі того ж обсягу даних [17].

Інтернет речей часто працює в умовах високої затримки та нестабільних каналів зв’язку. Завдяки підтримці різних рівнів якості обслуговування, збереженню інформації про сеанс і використанню постійних з’єднань MQTT забезпечує надійну доставку повідомлень навіть у таких умовах, що робить його одним із оптимальних протоколів для IoT-застосунків [17].

Безпека є критично важливою в мережах Інтернету речей, оскільки вони часто передають конфіденційні дані. MQTT підтримує шифрування Transport Layer Security (TLS) та Secure Sockets Layer (SSL), що забезпечує конфіденційність даних під час передачі. Крім того, він забезпечує механізми автентифікації та авторизації за допомогою облікових даних імені користувача/пароля або клієнтських сертифікатів, захищаючи доступ до мережі та її ресурсів [17].

Модель публікації-підписки MQTT забезпечує безперебійний двонаправлений зв'язок між пристроями. Клієнти можуть як публікувати повідомлення в темах, так і підписуватися на отримання повідомлень на певні теми, що забезпечує ефективний обмін даними в різноманітних екосистемах Інтернету речей без прямого зв'язку між пристроями. Ця модель

також спрощує інтеграцію нових пристроїв, забезпечуючи легку масштабованість [17].

MQTT дозволяє клієнтам підтримувати сесії з брокером з відстеженням стану, що дозволяє системі запам'ятовувати підписки та недоставлені повідомлення навіть після відключення. Клієнти також можуть вказати інтервал keep-alive під час з'єднання, який спонукатиме брокера періодично перевіряти стан з'єднання. Якщо з'єднання втрачено, брокер зберігає недоставлені повідомлення (залежно від рівня якості обслуговування) та намагається доставити їх, коли клієнт знову підключається. Ця функція забезпечує надійний зв'язок та зменшує ризик втрати даних через переривчасте з'єднання [17].

Системи Інтернету речей часто включають велику кількість пристроїв, що вимагає протоколу, здатного обробляти масштабні розгортання. Легка структура MQTT, низьке споживання пропускної здатності та ефективне використання ресурсів роблять його добре придатним для масштабних застосувань Інтернету речей. Шаблон публікації-підписки дозволяє MQTT ефективно масштабуватися, оскільки він розділяє відправника та одержувача, зменшуючи мережевий трафік та використання ресурсів. Крім того, підтримка протоколом різних рівнів якості обслуговування дозволяє налаштовувати доставку повідомлень на основі вимог застосування, забезпечуючи оптимальну продуктивність у різних сценаріях [17].

Системи IoT часто включають пристрої та програми, розроблені з використанням різних мов програмування. Широка підтримка мов MQTT забезпечує легку інтеграцію з кількома платформами та технологіями, сприяючи безперебійній комунікації та сумісності в різноманітних екосистемах Інтернету речей. [17].

Як зазначалось, протокол працює за шаблоном «Публікація-підписка» (згідно рисунку 2.1), де клієнт MQTT або публікує повідомлення в певній темі, або підписується на тему для отримання повідомлень, і все це керується

брокером MQTT, який забезпечує доставку повідомлень відповідно до заданих рівнів якості обслуговування [17].

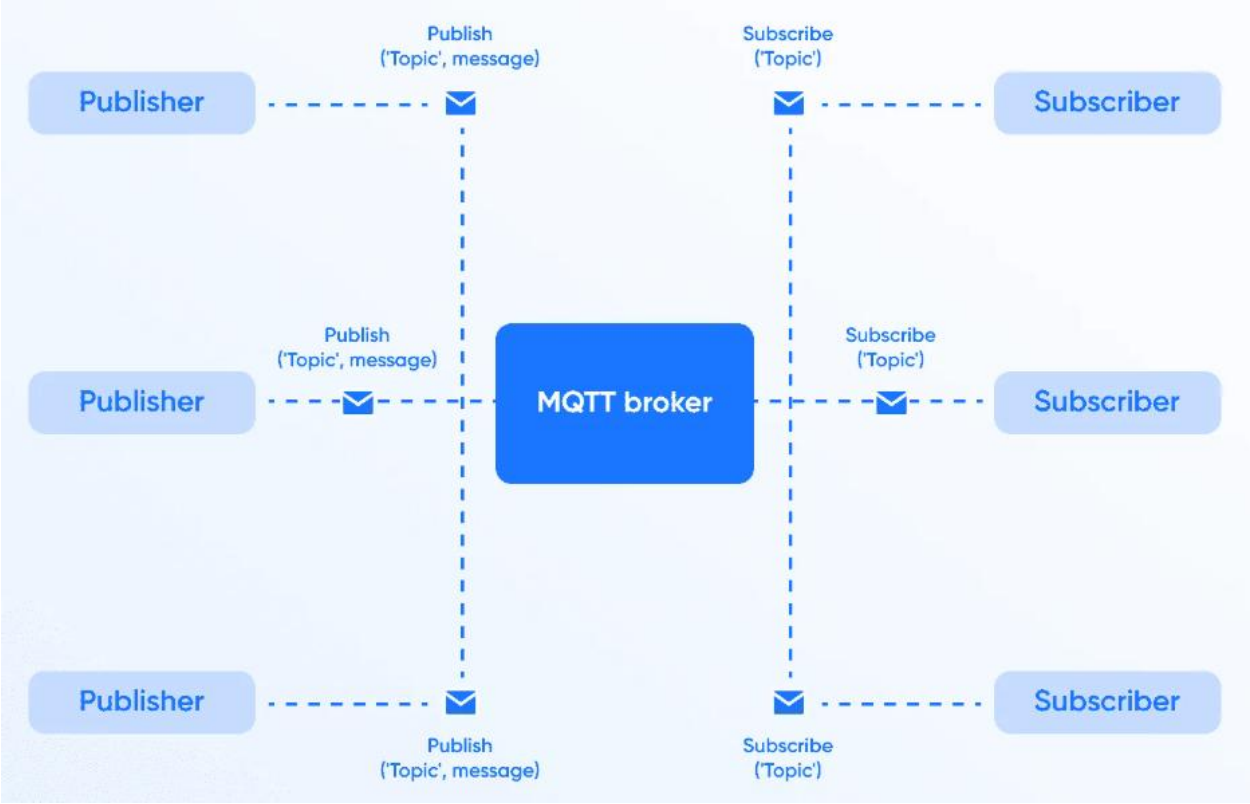


Рисунок 2.1 – Робочий процес брокера MQTT [16]

Будь-який додаток або пристрій, на якому працює клієнтська бібліотека MQT, є клієнтом MQTT. Наприклад, додаток для обміну миттєвими повідомленнями, який використовує MQTT, є клієнтом, різні датчики, які використовують MQTT для звітування даних, є клієнтом, а різні інструменти тестування MQTT також є клієнтом [17].

Брокер MQTT обробляє запити на підключення та відключення клієнтів, підписку та скасування підписки, а також маршрутизує повідомлення. Потужний брокер MQTT може підтримувати масові з'єднання та пропускну здатність повідомлень на рівні мільйонів, допомагаючи постачальникам послуг Інтернету речей зосередитися на бізнесі та швидко створювати надійні MQTT-додатки [17].

На рисунку 2.2 показано процес публікації/підписки на MQTT. Датчик температури підключається до сервера MQTT як клієнт і публікує дані про температуру в темі Temperature, а сервер отримує повідомлення та пересилає його клієнту [17].

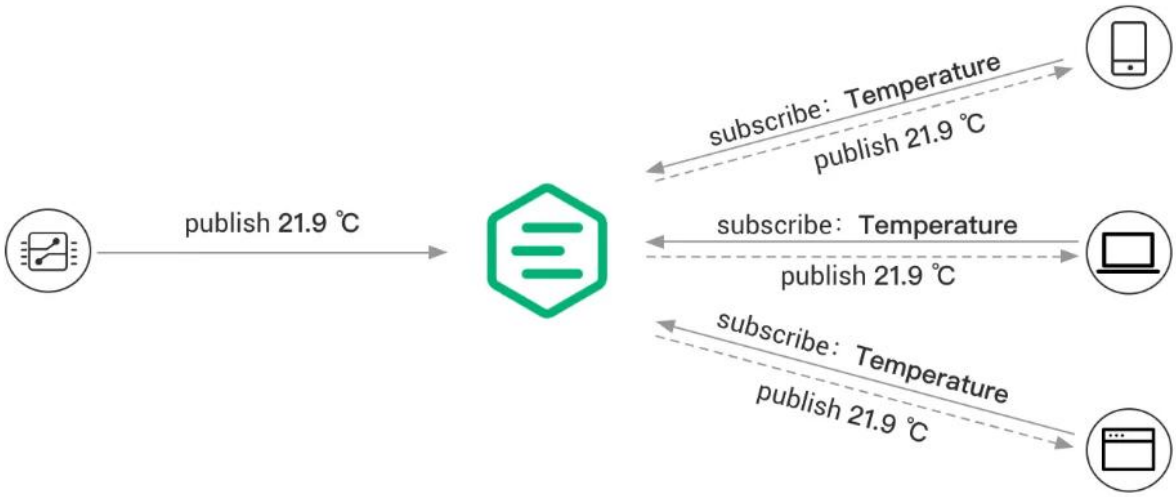


Рисунок 2.2 – Робочий процес брокера MQTT [17]

Варто зазначити, що MQTT забезпечує три види якості обслуговування (QoS) та гарантує надійність обміну повідомленнями в різних мережевих середовищах.

- QoS 0: Повідомлення доставляється не більше одного разу. Якщо клієнт наразі недоступний, він втратить це повідомлення.
- QoS 1: Повідомлення доставляється принаймні один раз.
- QoS 2: Повідомлення доставляється лише один раз [17].

2.2.2 DDS

DDS – це відкритий стандарт, який використовується переважно для програм реального часу. В першу чергу, це стандарт API та протокол проміжного програмного забезпечення, що зосереджений на забезпеченні сумісного, високопродуктивного та масштабованого обміну даними за

шаблоном публікації-підписки. Це єдиний відкритий стандарт, що використовується для обміну повідомленнями, який забезпечує підтримку унікальних потреб підприємства та систем реального часу. Він підтримує всі програми, що потребують обміну даними в режимі реального часу, такі як оборона, аерокосмічна галузь, моделювання, управління повітряним рухом та тестування. Це технологія для підключення програмного забезпечення, яка забезпечує безпечний обмін даними в режимі реального часу, швидку інтеграцію в промисловий Інтернет речей та розробку модульних програм [18].

Застосовуючи методологію публікації-підписки, CoAP в основному використовується для зв'язку M2M. Використовує багатоадресну розсилку для підвищення якості обслуговування програм (рисунок 2.3) [18].

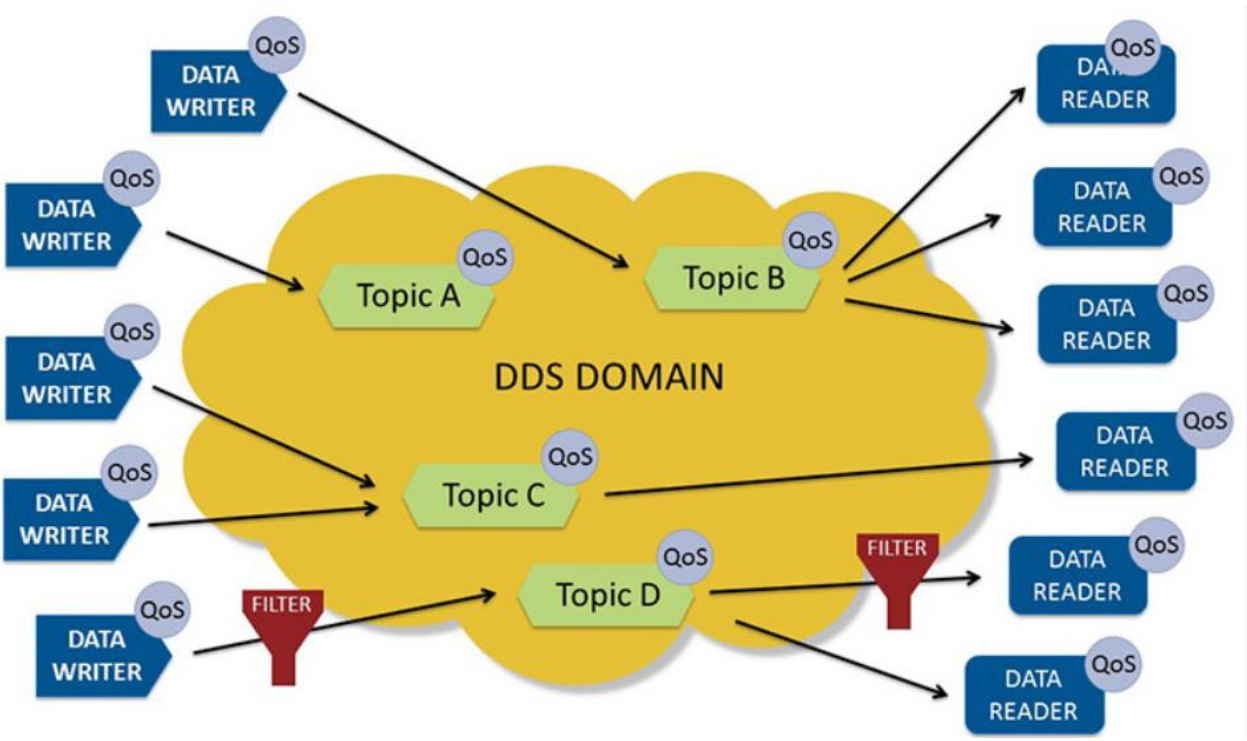


Рисунок 2.3 – Рівні якості обслуговування в протоколі DDS [19]

Протокол розробляється та управляється Групою управління об'єктами (Object Management Group). У протоколі DDS є два рівні, які називаються:

- DCPS
- DLRL

Термін DCPS – це скорочена форма від Data-Centric Publish-Subscribe (публікування-підписка, орієнтована на дані). Це стандартний інтерфейс прикладного програмування для рівня підписки/публікації, орієнтованого на дані та в режимі реального часу, який базується на темах. Основна мета цього рівня – доставка даних передплатникам [18].

Термін DLRL є абревіатурою від концепції Data Local Reconstruction Layer (Рівень локальної реконструкції даних). Це стандартний інтерфейс прикладного програмування для створення об'єктних представлень з колекції тем. Його метою є забезпечення інтерфейсу для функціональних можливостей DCPS. Це дозволяє безперешкодно обмінюватися розподіленою інформацією між пристроями з підтримкою Інтернету речей (IoT) [18].

DDS у сферах віртуалізації та центрів обробки даних охоплює як високорівневе використання програмного забезпечення, так і низькорівневі вдосконалення. Від обробки машин та віртуальних блоків для ефективного обміну даними до взаємодії обчислювальних ядер та віртуальних машин один з одним. DDS часто є ідеальним вибором для додаткових або вбудованих можливостей інтеграції. Використовуючи свої можливості багатоадресної розсилки, DDS можна використовувати в розподілених мережах для створення вигляду єдиної мережі на базі DDS [18].

У секторі мереж та телекомунікацій протокол DDS використовується для забезпечення та оптимізації зв'язку між різним мережевим обладнанням, таким як оптичні транспондери. Це компоненти, що забезпечують основи мобільного транспорту 5G. Ефективні системи для встановлення налаштувань, відстеження обладнання та впровадження оновлень також розробляються та розповсюджуються через DDS [18].

Даний протокол використовується зацікавленими сторонами та організаціями у сфері охорони здоров'я для забезпечення сумісного підключення даних для клінічних систем та медичних пристроїв. Основні застосування в секторі охорони здоров'я включають глобальні медичні установи з вимогами в режимі реального часу, забезпечення медичних

пристроїв для безпеки пацієнтів лікарень та інтеграцію систем клінічного прийняття рішень, таких як медичні планшети та комп'ютери [18].

Основне використання DDS у секторі оборони включає озброєння та управління, навігаційні системи, радіолокаційні системи та системи керування. Відомі агентства, такі як NASA, використовують протокол DDS у своїх системах керування запуском, переважно в системах збору даних та SCADA [18].

Проблеми, пов'язані зі зміною клімату, створюють потребу в значних інноваціях завдяки широкому поширенню промислових IIoT-рішень. Проблемні області вирішуються за допомогою існуючих технологій. Більше того, DDS впроваджується як організаціями, так і державами для досліджень та розробок. По суті, DDS знаходить своє застосування в таких додатках, як зберігання та управління, виробництво електроенергії, а також контроль та розподіл [18].

Протокол забезпечує безпечний та надійний обмін даними між програмами та пристроями в режимі реального часу. Важливість протоколу DDS в Інтернеті речей робить його актуальним для робототехніки, транспорту та промислової автоматизації. DDS працює за моделлю публікації-підписки, де пристрої, тобто видавці, обмінюються даними, пов'язаними з будь-якою темою, а пристрої, тобто передплатники, отримують дані, які їх цікавлять. DDS може сприяти зв'язку V2I та V2V. Крім того, він також може допомогти у зборі даних з датчиків, встановлених у системах безпеки та автономних транспортних засобах [18].

2.2.3 CoAP

Протокол CoAP (Протокол обмежених застосунків), — це легкий протокол інтернет-застосунків, розроблений для обмежених пристроїв в Інтернеті речей (IoT). Він дає змогу малопотужним пристроям, таким як датчики та носимі пристрої, ефективно обмінюватися даними з серверами,

хмарними сервісами та іншими вузлами IP-мереж, використовуючи мінімальні обчислювальні й мережеві ресурси. [20, 21].

Протокол CoAP має компактну базову специфікацію, яку можна розширити для отримання додаткової функціональності. Працюючи через UDP, він забезпечує модель взаємодії запит/відповідь, забезпечуючи сумісність між різними пристроями Інтернету речей [20, 21].

CoAP є високонадійним, з механізмами для забезпечення доставки повідомлень у складних мережевих середовищах, що робить його ідеальним для пристроїв Інтернету речей з обмеженим підключенням або живленням [20, 21].

CoAP використовує архітектуру RESTful (Representational State Transfer). Архітектура протоколу CoAP (рисунок 2.4) дотримується цього підходу, чітко звертаючись до ресурсів через URL-адреси. Використовуючи такі методи, як GET (отримання даних), POST (надсилання нових даних), PUT (оновлення даних) та DELETE (видалення даних), ці ресурси можуть бути цільовими. Це дозволяє CoAP забезпечувати плавний, ефективний та надійний зв'язок навіть у високообмежених середовищах Інтернету речей. Як результат, він підтримує створення гнучких та масштабованих архітектур, що відповідають вимогам сучасних систем Інтернету речей [20, 21].

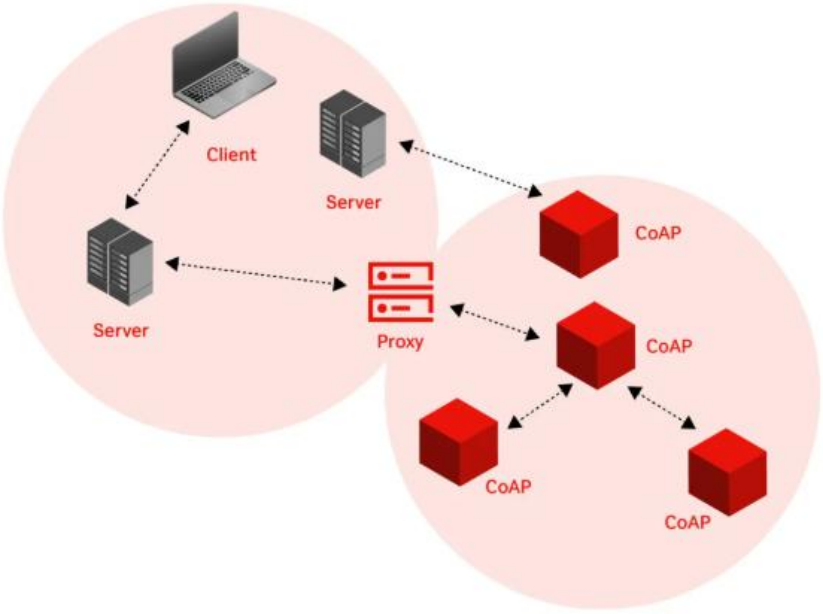


Рисунок 2.4 – Архітектура протоколу CoAP [21]

CoAP реалізує низку функціональних механізмів, які дозволяють забезпечити надійний та гнучкий обмін даними в динамічних та ненадійних IoT-мережах. До ключових з них належать вбудований механізм виявлення ресурсів, підтримка асинхронної взаємодії, система ідентифікації повідомлень та підтверджувані повідомлення, що разом компенсують недоліки використання ненадійного транспорту UDP [20].

У протоколі CoAP передбачено вбудований механізм виявлення ресурсів (service discovery), який дає змогу вузлам мережі отримувати інформацію про доступні ресурси на інших пристроях без попереднього знання їхньої конфігурації. Такий підхід є особливо актуальним для мереж Інтернету речей, де склад мережі змінюється динамічно, а пристрої можуть довільно приєднуватися та залишати мережу [20].

CoAP підтримує асинхронну модель обміну повідомленнями, яка дозволяє відправнику не блокуватися в очікуванні відповіді від віддаленого вузла. Це є критично важливим для пристроїв Інтернету речей, що працюють з переривчастим з’єднанням, переходять у режими енергозбереження або мають обмежені обчислювальні ресурси [20].

У такій моделі вузол може сформувати та надіслати запит, після чого продовжити виконання інших завдань, а отриману із затримкою відповідь обробити у зручний момент часу. Асинхронність дає змогу зменшити час активної роботи радіомодуля та оптимізувати використання енергоресурсів, що є ключовою вимогою для більшості IoT-сенсорів і вузлів [20].

Для коректної обробки асинхронних відповідей протокол CoAP використовує ідентифікатор повідомлення (Message ID), який включається до кожного пакета. Цей ідентифікатор дозволяє вузлу-відправнику зіставляти отримані відповіді з відповідними запитами, навіть якщо в мережі одночасно циркулює кілька запитів або відбуваються повторні передачі [20].

Додатково застосовуються токени (Token), які надають можливість розрізняти окремі транзакції на рівні прикладної логіки і не залежать від конкретного транспортного з’єднання. Сукупне використання Message ID та токенів забезпечує стійкість до дублювання пакетів, повторних передач і дозволяє підтримувати коректний стан взаємодії між вузлами навіть у складних мережевих умовах [20].

Надійність доставки в CoAP забезпечується за рахунок розподілу повідомлень на підтверджувані та непідтверджувані. Для підтверджуваних повідомлень відправник очікує отримання спеціального пакета-підтвердження (Acknowledgement, ACK), який свідчить про успішне надходження даних до одержувача [20].

У разі відсутності підтвердження протягом встановленого інтервалу часу запускається механізм повторної передачі, що може супроводжуватися збільшенням інтервалів між спробами відповідно до налаштованих тайм-аутів. Така схема дозволяє компенсувати втрати пакетів, характерні для бездротових та інших ненадійних каналів зв’язку, і забезпечує гарантовану доставку критично важливих повідомлень [20].

Поєднання механізму виявлення ресурсів, асинхронної взаємодії, чіткої ідентифікації повідомлень і підтверджуваних повідомлень дає змогу протоколу CoAP забезпечувати надійний обмін даними поверх ненадійного

UDP-транспорту. Пристрої можуть автоматично виявляти ресурси одне одного, ініціювати взаємодію без жорсткої часової синхронізації та гарантувати доставку важливих повідомлень навіть у разі втрат і затримок у мережі [20].

CoAP використовує Datagram Transport Layer Security (DTLS) для захисту зв'язку, захищаючи пристрої Інтернету речей від несанкціонованого доступу та витоків даних. DTLS забезпечує шифрування та автентифікацію, що робить CoAP придатним для використання в чутливих програмах, таких як охорона здоров'я та промисловий Інтернет речей [20].

2.2.4 AMQP

Розроблений для забезпечення безпечної та надійної передачі даних між різними платформами та мовами програмування, AMQP – це відкритий стандарт для обміну повідомленнями між комп'ютерними системами. Як протокол черги повідомлень, AMQP визначає, як повідомлення маршрутизуються, зберігаються та доставляються між відправниками та одержувачами. Це гарантує, що повідомлення не втрачаються, навіть у разі збоїв або затримок [22].

Протокол розширеної черги повідомлень використовує структуровану архітектуру для надійної та контрольованої передачі повідомлень між системами. Ключовими компонентами є продуцент, брокер, біржа та споживач (рисунок 2.5) [22].

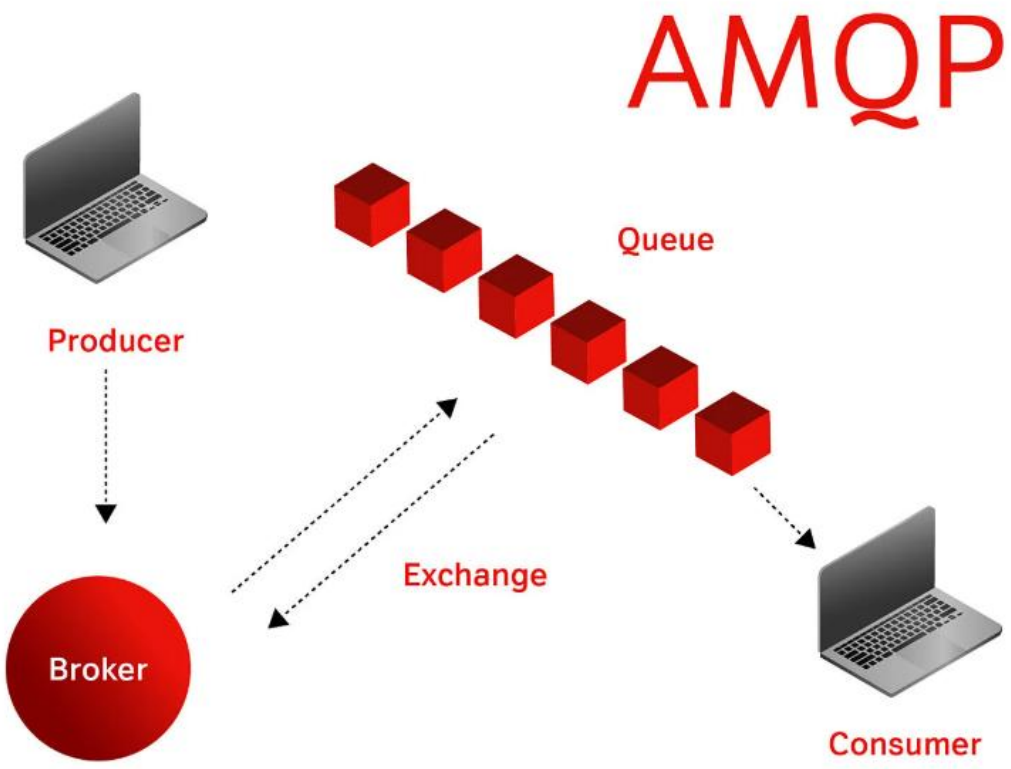


Рисунок 2.5 – Архітектура протоколу AMQP [22]

Принцип роботи протоколу AMQP:

Потік повідомлень починається з джерела (producer). Цей додаток або сервіс генерує повідомлення та надсилає його в систему. Однак повідомлення не передається безпосередньо одержувачу, а спочатку пересилається брокеру.

Брокер виступає центральним елементом у системі AMQP та керує всім трафіком повідомлень. Усередині брокера повідомлення досягає біржі .

Біржа, на основі попередньо визначених правил, оцінює, куди слід направити повідомлення. Ці правила встановлюються за допомогою прив'язок, які з'єднують біржу з однією або кількома чергами. Повідомлення тимчасово зберігаються в черзі, доки відповідний споживач не отримає їх .

Зрештою, споживачем є додаток, який отримує та обробляє повідомлення.

Спосіб, у який біржа розповсюджує повідомлення, залежить від типу використовуваної біржі:

- Прямий обмін: пересилає повідомлення лише тоді, коли ключ маршрутизації точно збігається з ключем зв'язування.
- Розподілений обмін: Надсилає кожне повідомлення до всіх підключених черг, незалежно від ключа маршрутизації.
- Обмін темами: використовує підстановлювальні символи в ключі маршрутизації для маршрутизації повідомлень на основі тем або категорій.
- Обмін заголовками: маршрутизує повідомлення на основі атрибутів у заголовку повідомлення, а не ключа маршрутизації [22].

Така архітектура робить стандарт AMQP особливо надійним, оскільки відправники та одержувачі не повинні бути активними одночасно. Повідомлення надійно зберігаються та пересилаються, як тільки цільова система готова. Як результат, протокол черги повідомлень забезпечує безпечний, відстежуваний та гнучкий зв'язок навіть у високорозподілених та складних IT-середовищах [22].

Стандарт AMQP пропонує комплексні функції безпеки для надійного передавання повідомлень.

Зв'язок між системами може бути зашифрований за допомогою протоколу TLS (Transport Layer Security), що гарантує захист повідомлень від несанкціонованого доступу.

Крім того, AMQP використовує протокол простої автентифікації та рівня безпеки (SASL) для унікальної автентифікації користувачів. Залежно від вимог використовуються різні методи, такі як паролі або сертифікати.

Детальний контроль доступу дозволяє точно вказати, яким користувачам або системам дозволено надсилати, отримувати або керувати повідомленнями, що дозволяє цілеспрямовано призначати права.

Базова інфраструктура також має бути захищена, наприклад, за допомогою брандмауерів та регулярного моніторингу [22].

Використання AMQP:

- AMQP дозволяє корпоративним застосункам, системам і базам даних надійно та безпечно обмінюватися даними. Це особливо корисно в гетерогенних середовищах, де застосунки побудовані на різних платформах і технологіях, які повинні безперебійно працювати разом [23].
- AMQP використовується в сценаріях, що вимагають обробки даних у реальному часі, таких як потокова аналітика, де дані необхідно збирати, обробляти та аналізувати негайно. Протокол забезпечує надійну доставку повідомлень навіть під час збоїв мережі або затримок обробки [23].
- AMQP може розподіляти завдання або робочі навантаження між кількома робочими процесами або службами, допомагаючи збалансувати навантаження та підвищити загальну ефективність і швидкість реагування системи. Ця функція особливо корисна в хмарних обчисленнях та розподілених обчислювальних середовищах з непередбачуваними робочими навантаженнями [23].
- AMQP підтримує асинхронні шаблони зв'язку, що дозволяє програмам надсилати та отримувати повідомлення без блокування операцій. Цей тип зв'язку є критично важливим для програм, які потребують високої пропускної здатності та низької затримки, оскільки дозволяє їм залишатися чутливими під час очікування повідомлень [23].
- У застосунках IoT AMQP використовується для збору даних з різних датчиків і пристроїв та їх передачі до систем обробки або хмарних сервісів. Його здатність працювати в обмежених мережах, а також безпечний і надійний обмін повідомленнями роблять його чудовим вибором для сценаріїв IoT [23].
- Використовується у фінансовій галузі для обробки транзакцій, систем торгівлі в реальному часі та обробки платежів. Його надійність та підтримка транзакційних повідомлень забезпечують цілісність та узгодженість фінансових транзакцій [23].

- Протокол можна використовувати для впровадження систем сповіщень, де сповіщення, повідомлення або електронні листи надсилаються користувачам або системам у відповідь на певні події чи умови. Його надійний обмін повідомленнями гарантує доставку сповіщень, навіть якщо одержувач тимчасово недоступний [23].
- У сфері охорони здоров'я AMQP забезпечує безпечний та надійний зв'язок між різними системами, такими як записи пацієнтів, виставлення рахунків та діагностичне обладнання. Він гарантує безпечну передачу та обробку конфіденційних даних відповідно до норм [23].

2.2.5 HTTP

Протокол HTTP або протокол передачі гіпертексту — це протокол прикладного рівня для передачі гіпермедійних документів, таких як HTML, через Інтернет. Він є основою передачі даних, що дозволяє веббраузерам (клієнтам) та вебсерверам ефективно взаємодіяти за допомогою моделі запит-відповідь (згідно рисунку 2.6) [24].

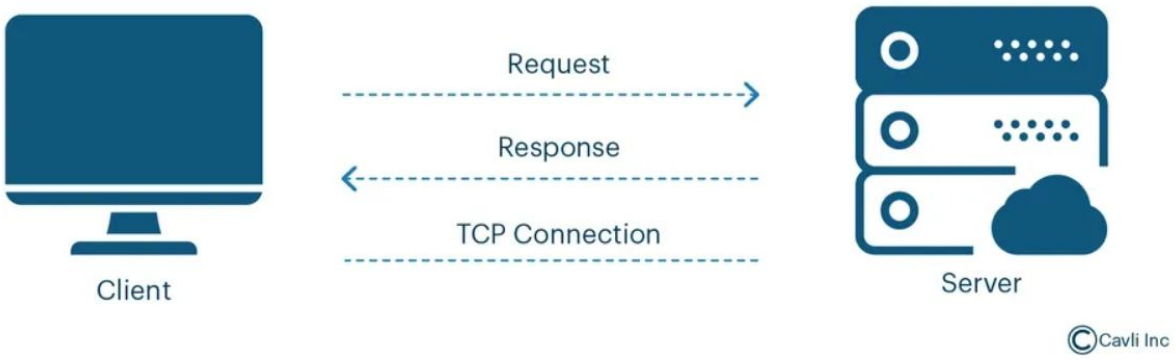


Рисунок 2.6 –Протокол HTTP [24]

У цій моделі клієнт (зазвичай веб-браузер) ініціює процес, надсилаючи HTTP-запит на сервер. Цей запит містить метод запиту (GET, POST, PUT, DELETE тощо), цільовий URI (уніфікований ідентифікатор ресурсу, наприклад, URL-адреса), заголовки та необов'язкове тіло запиту [36].

Сервер отримує запит і обробляє його на основі запитуваного методу та ресурсу. Це може включати отримання даних з бази даних, виконання серверних скриптів або виконання інших операцій [36].

Після обробки запиту сервер надсилає HTTP-відповідь клієнту. Відповідь містить код стану (наприклад, 200 OK, 404 Not Found), заголовки відповіді та необов'язкове тіло відповіді, що містить запитувані дані або контент [36].

Клієнт отримує відповідь сервера та обробляє її відповідно. Наприклад, якщо відповідь містить HTML-сторінку, браузер відобразить її. Якщо це зображення або інший медіафайл, браузер відобразить або обробить його відповідним чином [36].

Протокол HTTP не зберігає стан, тобто кожен запит є незалежним і не зберігає інформацію про попередні взаємодії [36]. Він підтримує різні методи:

- GET: отримання даних із зазначеного ресурсу. Не повинен мати побічних ефектів і зазвичай використовується для отримання веб-сторінок, зображень тощо [36].
- POST: надсилання даних для обробки певним ресурсом. Підходить для надсилання форм, завантаження файлів та створення нових ресурсів [36].
- PUT: оновлення або створення ресурсу на сервері. Він замінює весь ресурс даними, наданими в тілі запиту [36].
- PATCH: подібний до PUT, але використовується для часткових змін ресурсу. Він оновлює певні поля ресурсу, а не замінює весь ресурс [36].
- DELETE: видалення вказаного ресурсу з сервера [36].
- HEAD: подібний до GET, але отримує лише заголовки відповіді, корисно для перевірки властивостей ресурсу без передачі повного вмісту [36].

- OPTIONS: отримання доступних для ресурсу параметрів зв'язку, включаючи підтримувані методи та заголовки [36].
- TRACE: налагодження, щоб відобразити отриманий запит назад клієнту, хоча рідко використовується через міркування безпеки [36].
- CONNECT: Використовується для встановлення тунелю до сервера через HTTP-проксі, зазвичай використовується для SSL/TLS-з'єднань [36].

Безпечний зв'язок забезпечується через HTTPS, який включає протоколи шифрування, такі як TLS (Transport Layer Security), для забезпечення цілісності та конфіденційності даних під час передачі [24].

Оскільки HTTPS (рисунок 2.7) пропонує додатковий рівень безпеки та довіри, він забезпечує спосіб захисту даних користувачів. Це дозволяє компаніям (власникам веб-сайтів) заслужити довіру користувачів. Впровадження HTTPS також може позитивно вплинути на пошукову оптимізацію сайту, підвищуючи його рейтинг на сторінках результатів пошуку [38].

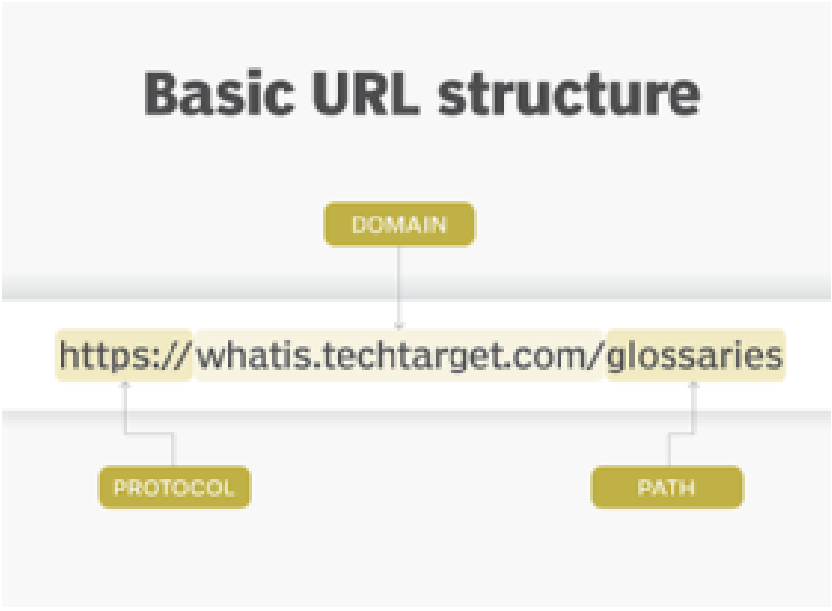


Рисунок 2.7 – HTTPS [38]

На рисунку 2.8 представлені три версії протоколу передачі гіпертексту (HTTP) — HTTP, HTTP/2 та HTTP/3.

HTTP/1.1 використовує текстовий формат повідомлень поверх TCP-з’єднання та модель «один запит — одна відповідь» на одному каналі. [37, 38].

HTTP/2 зберігає ту саму логіку роботи HTTP (методи, коди стану, структуру запитів і відповідей), але використовує бінарний формат кадрів і дозволяє передавати кілька запитів одночасно в одному TCP-з’єднанні. Завдяки стисненню заголовків і пріоритезації потоків завантаження сторінок з великою кількістю дрібних ресурсів стає значно швидшим. [37, 38].

HTTP/3 реалізує HTTP на транспортний протокол QUIC, що усуває проблему блокування на рівні TCP, поєднуючи в один етап встановлення з’єднання та налаштування шифрування (TLS), і краще переносить втрату пакетів та зміну мережі (наприклад, перехід з Wi-Fi на мобільний інтернет), що зменшує затримки та підвищує стабільність з’єднання [37, 38].

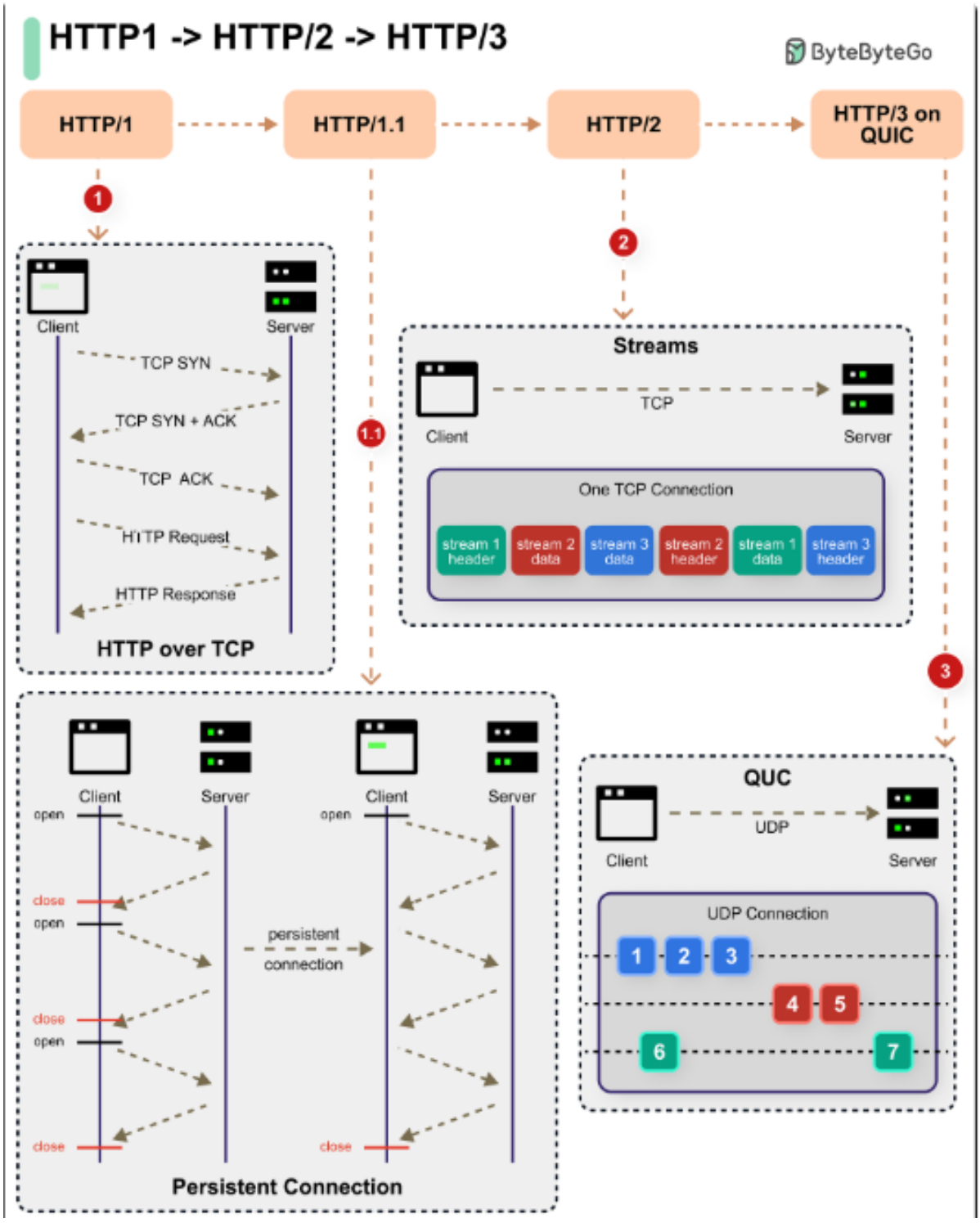


Рисунок 2.8 – Версії протоколу HTTP [37]

2.3 Проблеми та обмеження протоколів рівня застосунків

Проблеми та обмеження протоколів рівня застосунків в IoT зумовлені поєднанням трьох факторів: обмеженості ресурсів пристроїв, нестабільності мереж та зростаючих вимог до безпеки й масштабованості.

Багато IoT-вузлів мають низьку обчислювальну потужність, обмежений обсяг пам’яті та працюють від батареї, що робить для них критичними розмір заголовків, частоту обміну та складність стека протоколів. Протоколи з текстовими заголовками й великою службовою частиною (наприклад, HTTP, AMQP) можуть суттєво збільшувати енергоспоживання і затримки порівняно з легковагими бінарними протоколами (MQTT, CoAP) [15, 25, 26].

IoT-мережі часто працюють у середовищах із високим рівнем втрат, змінною затримкою та обмеженою пропускною здатністю (бездротові сенсорні мережі, LPWAN тощо). Хоча протоколи типу MQTT та CoAP включають механізми повторної передачі, підтверджень та QoS, у реальних умовах забезпечення гарантованої доставки й передбачуваних затримок залишається складним завданням [15, 25, 26].

Ситуацію ускладнює гетерогенність мережної інфраструктури: різні сегменти можуть використовувати різні технології доступу й транспортні протоколи, що потребує застосування шлюзів і проксі для протокольної конвергенції. Це збільшує затримки, ускладнює діагностику та створює додаткові точки відмови [15, 25, 26].

Брокер-орієнтовані протоколи (MQTT, AMQP) створюють централізовані точки концентрації трафіку, що спрощує керування, але водночас породжує проблеми масштабованості й відмовостійкості. Зі зростанням кількості пристроїв брокери можуть ставати «вузьким місцем» системи, вимагати горизонтального масштабування, балансування навантаження й складних механізмів кластеризації [15, 25, 26].

Навіть у децентралізованих рішеннях (наприклад, DDS) масштабування супроводжується зростанням обсягу службового трафіку для виявлення учасників і узгодження QoS-політик, що може бути критичним для обмежених мереж [15, 25, 26].

Використання різних моделей даних, форматів повідомлень та протоколів у межах однієї системи призводить до проблем інтероперабельності. Навіть за однакового протоколу (наприклад, MQTT)

відсутність єдиних семантичних угод щодо структури payload (JSON, CBOR, власні формати) ускладнює інтеграцію рішень різних вендорів [15, 25, 26].

Для подолання цих обмежень потрібні додаткові рівні абстракції (шлюзи, платформи управління даними, онтології), що збільшує складність архітектури й створює залежність від конкретних платформних рішень [15, 25, 26].

Забезпечення конфіденційності, цілісності й автентифікації (TLS, DTLS, DDS Security тощо) неминуче збільшує обчислювальні витрати, обсяг переданих даних і затримки. Для ресурсно-обмежених пристроїв це може бути критичним: повноцінні криптографічні механізми скорочують час автономної роботи й роблять частину протоколів практично непридатними без додаткових оптимізацій [15, 25, 26].

При цьому багато протоколів покладаються на безпеку нижчих рівнів (наприклад, MQTT — на TLS), не маючи розвинених власних засобів керування доступом і авторизації, що підвищує роль правильної конфігурації інфраструктури й збільшує ризики людських помилок [15, 25, 26].

Реальне впровадження протоколів рівня застосунків у великих IoT-системах вимагає налаштування численних параметрів: рівнів якості обслуговування, розмірів черг, тайм-аутів, політик повторної передачі, параметрів шифрування та авторизації. Неправильний вибір цих параметрів може призвести до перевантаження мережі, збільшення затримок або, навпаки, до втрати критичних повідомлень [15, 25, 26].

2.4 Порівняльний аналіз протоколів за параметрами продуктивності, енергоефективності та безпеки

Проаналізувавши протоколи рівня застосунків в попередніх пунктах розділу можна зробити наступну таблицю з їх порівняльним аналізом за п’ятьма параметрами.

Таблиця 2.1 – Порівняльний аналіз протоколів рівня застосунків

Протокол	Модель	Транспортний протокол	Експлуатаційні витрати	Енергоефективність	Типові механізми безпеки
MQTT	Publish/ subscribe	TCP	Низькі для малих повідомлень	Висока на вузьких каналах	SSL, аутентифікація на рівні брокера
DDS	Publish/ subscribe	TCP/UDP	Вищі, складна реалізація	Здебільшого не для обмежених вузлів	SSL/DTLS
CoAP	Publish/ subscribe	UDP	Дуже низькі	Дуже висока для обмежений вузлів	DTLS
AMQP	Broker - based messaging	TCP	Вищі через складні структури	Орієнтація на потужні пристрої	TLS, SASL
HTTP(S)	Запит - відповідь	TCP	Високі через заголовки	Нижча на обмежених вузлах	SSL/HTTPS

2.5 Висновки з розділу 2

У другому розділі було системно обґрунтовано місце протоколів рівня застосунків у загальній архітектурі систем Інтернету речей та показано, що саме вони визначають логіку взаємодії між IoT-пристроями, шлюзами, платформами та прикладними сервісами. Розкрито, що протоколи прикладного рівня відповідають не лише за формат і структуру переданих повідомлень, а й за моделі комунікації (клієнт–сервер, публікація–підписка), механізми забезпечення надійності, підтримку параметрів якості обслуговування, засоби аутентифікації та шифрування.

У межах аналізу було визначено основні протоколи рівня застосунків, що використовуються в IoT. Зокрема, розглянуто протоколи MQTT, DDS, CoAP, AMQP, HTTP.

Порівняльний аналіз за параметрами продуктивності, енергоефективності та безпеки дав змогу продемонструвати, що вибір

протоколу рівня застосунків завжди є компромісом між затримками, пропускною здатністю, накладними витратами, надійністю доставки та складністю реалізації захисних механізмів. Було показано, що, наприклад, MQTT добре підходить для малоресурсних пристроїв і сценаріїв телеметрії завдяки моделі «публікація–підписка» та гнучкій підтримці якості обслуговування, тоді як CoAP краще інтегрується з REST-архітектурою та constrained-середовищами, а застосування HTTP є доцільним у випадках, коли пріоритетними є сумісність із наявною веб-інфраструктурою та простота інтеграції. Варто зазначити, що питання безпеки не можуть розглядатися ізольовано від продуктивності, оскільки використання криптографічних засобів і протоколів захищеного транспортних протоколів прямо впливає на енергоефективність і затримки в IoT-мережах.

Аналіз проблем і обмежень протоколів прикладного рівня показав низку системних викликів, характерних для сучасних IoT-екосистем. Серед основних проблем варто відзначити фрагментованість стандартів і профілів реалізації, що ускладнює взаємодію пристроїв від різних виробників; недостатній рівень вбудованих засобів безпеки в надлегких протоколах; залежність продуктивності від стану мережевого середовища; а також труднощі масштабованого керування сесіями, підписками та оновленнями конфігурацій у великих розподілених системах.

Подолання цих обмежень вимагає не лише вдосконалення самих протоколів, а й комплексного підходу до побудови архітектури IoT-рішень — поєднання правильного вибору протоколу з ефективною організацією мережевої взаємодії, платформних рішень і механізмів безпеки.

3 ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ОБМІНУ ДАНИМИ ПРОТОКОЛІВ РІВНЯ ЗАСТОСУНКІВ

3.1 Постановка задачі та методика дослідження ефективності протоколів

У роботі проводиться експериментальне порівняння п’яти протоколів прикладного рівня, що використовуються в IoT-системах: MQTT, DDS, AMQP, CoAP та HTTP, у рамках єдиного програмно-апаратного середовища.

Мета дослідження – експериментально оцінити ефективність обраних протоколів за двома базовими показниками якості обміну даними:

- затримкою доставки повідомлень (end-to-end latency);
- надійністю передачі, вираженою коєфіцієнтом успішної доставки пакетів (Packet Delivery Ratio, PDR).

Затримка доставки визначається як час від моменту формування повідомлення на стороні клієнта до моменту його отримання та обробки на логічному шлюзі. У практиці аналізу мереж end-to-end затримка вимірюється як різниця між часовою міткою, вставленою відправником, та часом отримання пакета отримувачем. У роботі для кожного сценарію визначаються: середня, а також мінімальна та максимальна затримки [39].

Коєфіцієнт успішної доставки пакетів (Packet Delivery Ratio, PDR) обчислюється як: N_{recv} (кількість унікальних прийнятих повідомлень) поділити на N_{sent} (кількість фактично надісланих повідомлень) [39].

PDR характеризує надійність протоколу та його стійкість до втрат при зростанні навантаження.

Згідно з узагальнюючими оглядами, саме end-to-end затримка та PDR є базовими метриками при оцінюванні продуктивності протоколів і мереж, доповнюючись іншими параметрами (пропускна здатність, енергоспоживання тощо) залежно від задачі [39].

Дослідження буде проводитись в середовищі Node-RED, де створюються окремі потоки для кожного з досліджуваних протоколів, які

приймають вхідні повідомлення, розшифровують дані, фіксують момент їх отримання, обчислюють затримку на основі часової мітки відправлення та записують результати. Для забезпечення порівнянності результатів для всіх протоколів використовується єдина структура повідомлень у форматі JSON [40].

Сценарії навантаження формуються таким чином, щоб відобразити типові режими роботи IoT-систем. Для кожного протоколу запускається серія експериментів із різною кількістю одночасних клієнтів, різними інтервалами передавання повідомлень і розмірами корисного навантаження. Кожен логічний клієнт генерує фіксовану кількість повідомлень у кожному сценарії, що дає змогу накопичити статистично достовірні вибірки для подальшого аналізу. За результатами серій вимірювань формуються масиви значень затримки та показників PDR, які опрацьовуються засобами табличних процесорів на основі чого будуються порівняльні таблиці.

3.2 Розробка тестового середовища для моделювання обміну даними

Дослідження проводиться на уніфікованому тестовому стенді, реалізованому на персональному комп’ютері (ОС Windows 10). Стенд включає такі компоненти:

- Node-RED – графічне середовище потокової обробки даних, яке використовується як логічний шлюз IoT-системи (рисунок 3.1) [40].
- MQTT - брокер (Mosquitto) для обслуговування MQTT-клієнтів.
- AMQP - брокер (RabbitMQ) для реалізації черг повідомлень протоколу AMQP.
- Підсистема CoAP для Node-RED (вузли node-red-contrib-coap) для приймання CoAP-повідомлень.
- Компоненти інтеграції з DDS (адаптер, що передає DDS-повідомлення в Node-RED через TCP).
- HTTP-endpoint у Node-RED для прийому та обробки HTTP-запитів.

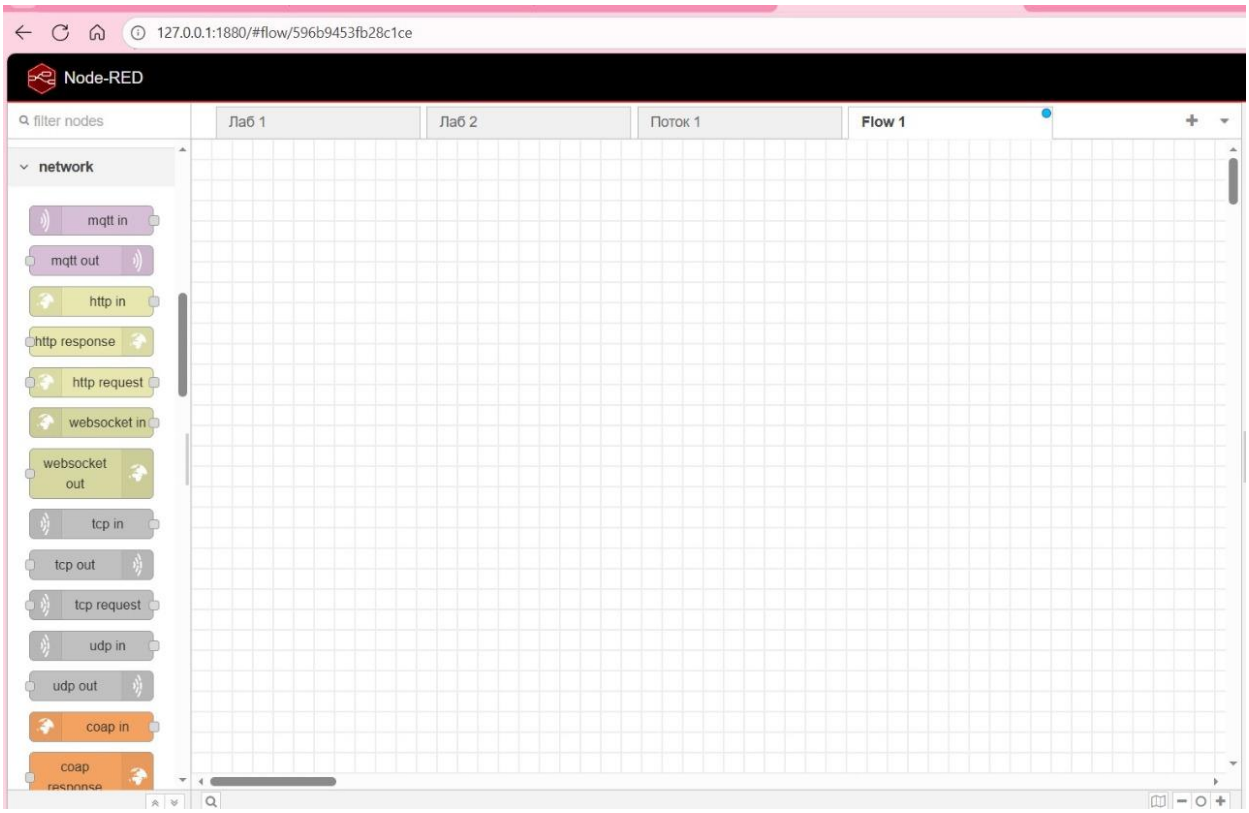


Рисунок 3.1 – Середовище Node-RED

Node-RED забезпечує побудову потоків, які з’єднують вхідні вузли протоколів (mqtt in, coap in, http in, amqp in, адаптер DDS) з вузлами обробки (JSON-декодування, функції обчислення затримки, логування результатів). Такий підхід дозволяє реалізувати вимірювання даних для всіх протоколів, змінюючи лише транспортний рівень [40].

Кожен потік складається з таких етапів:

1. прийом повідомлення відповідним вузлом (mqtt in, тощо);
2. декодування JSON-повідомлення;
3. фіксація часу отримання та обчислення затримки;
4. формування уніфікованого запису результату;
5. збереження результатів у CSV-файл.

3.3 Приклад реалізації обміну даними та аналіз роботи протоколів

Спершу у командному рядку персонального комп’ютера вводиться команда згідно рисунку 3.2.

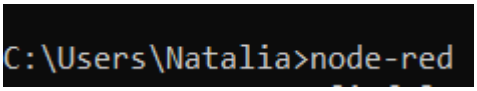


Рисунок 3.2 – Запуск Node.js

Наступним кроком у браузері вводиться посилання *http://127.0.0.1:1880* для відкриття робочого середовища (рисунок 3.3).

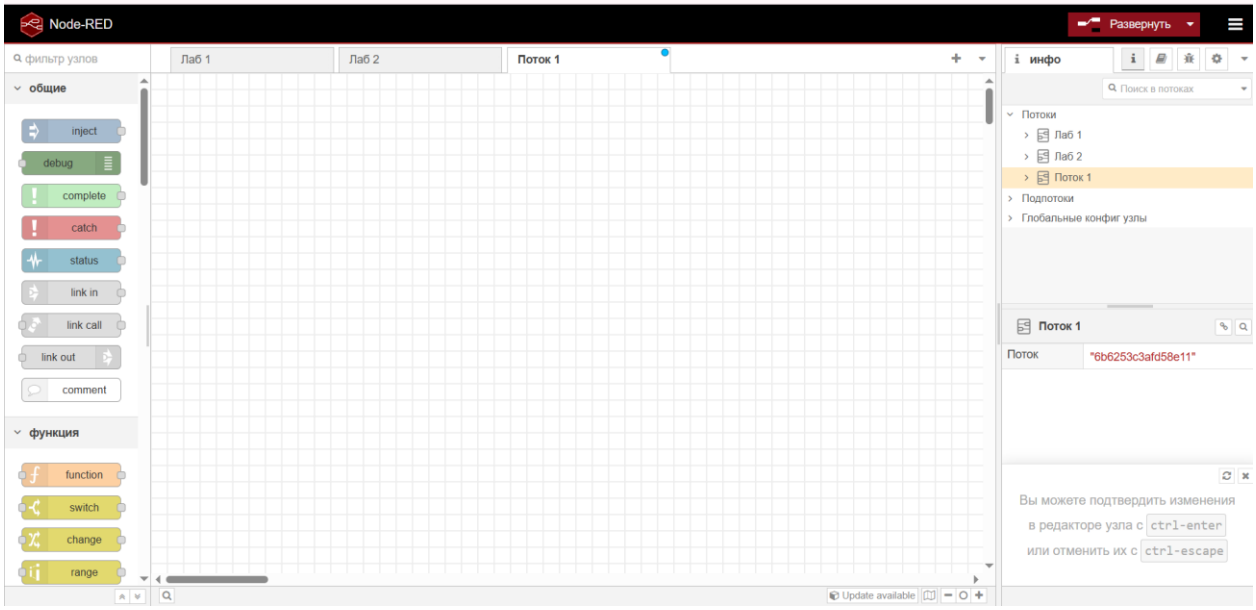


Рисунок 3.3 – Робоче середовище Node-RED

На рис. 3.4 наведено схему потоку для протоколу MQTT. Потік забезпечує прийом повідомлень від MQTT-клієнтів, декодування JSON-даних, обчислення end-to-end затримки та запис результатів у файл для подальшого аналізу.

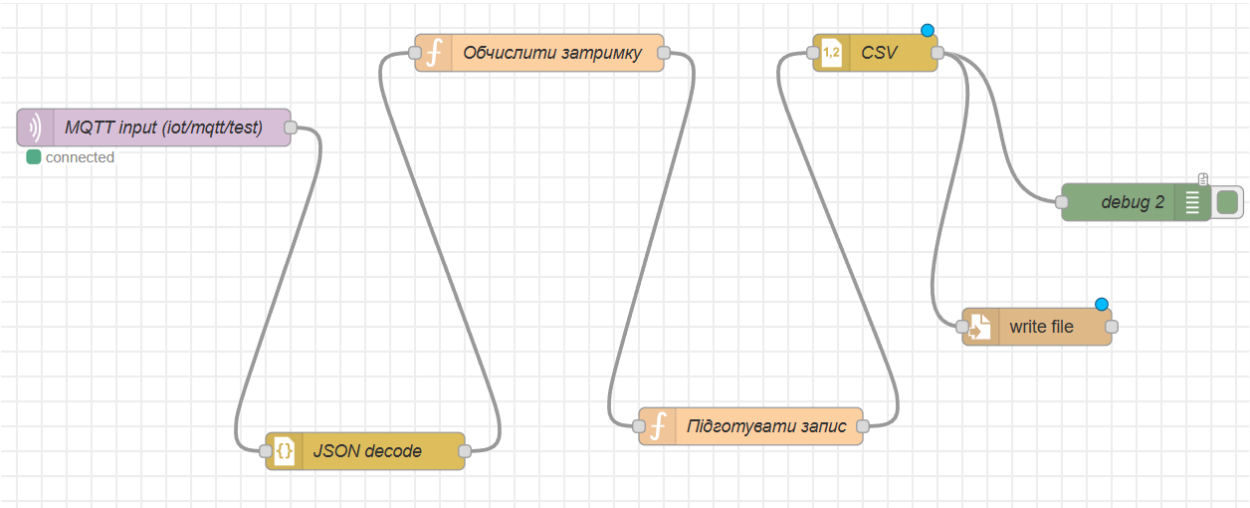


Рисунок 3.4 –MQTT

Вузол MQTT input (iot/mqtt/test) – це перший елемент потоку, який виконує підписку на відповідний MQTT-топік і слугує точкою входу повідомлень у Node-RED.

Основні параметри конфігурації зображені на рисунку 3.5.

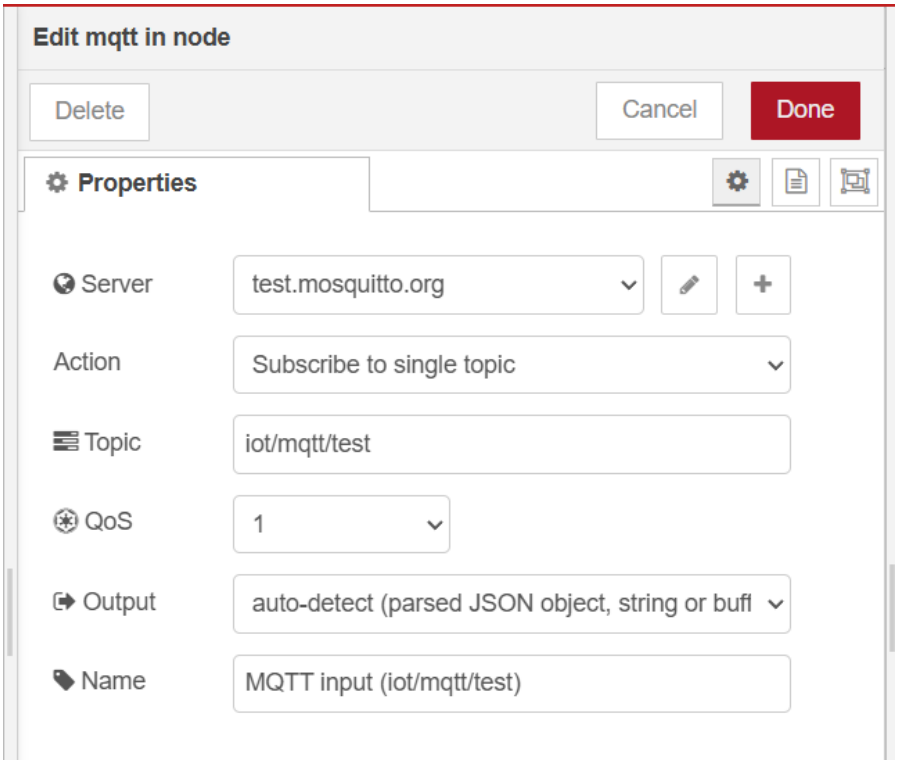
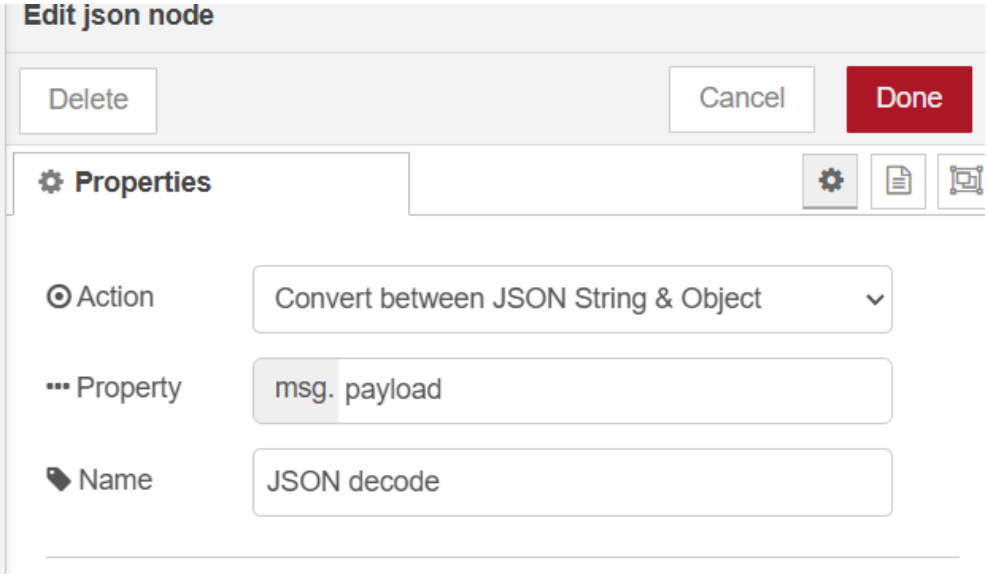


Рисунок 3.5 – Параметри конфігурації вузла MQTT

Другим етапом є вузол json, який перетворює текстове JSON-представлення у внутрішній об’єкт JavaScript. Налаштування вузла зображено на рисунку 3.6.



- Рисунок 3.6 – Параметри конфігурації вузла json

Наступний елемент – функціональний вузол function, у якому реалізовано обчислення end-to-end затримки доставки повідомлення. Вузол зчитує часову мітку відправлення, додану клієнтом (timestamp_sent), фіксує поточний час на шлюзі та обчислює різницю між ними.

Вміст функції:

```
import time, random
import paho.mqtt.client as mqtt
import json

client = mqtt.Client()
client.connect("localhost", 1883, 60)

PROTOCOL = "MQTT"
CLIENT_ID = 1
```

```
for i in range(100):  # 100 повідомлень
    timestamp_sent = int(time.time() * 1000)  # мс
    payload = {
        "protocol": PROTOCOL,
        "client_id": CLIENT_ID,
        "timestamp_sent": timestamp_sent,
        "payload": "test"
    }
    client.publish("iot/mqtt/test", json.dumps(payload), qos=1)
    time.sleep(1.0)  # 1000 мс
```

Щоб спростити подальший експорт результатів у CSV, у наступному функціональному вузлі формується уніфікований об’єкт, який містить тільки необхідні поля.

Вміст функції:

```
import random, csv
with open("dds_results_standard.csv", "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
writer.writerow(["protocol","client_id","latency_ms","timestamp_sent","timestamp_received"])
    for client_id in range(1, 101):      # 100 клієнтів
        for i in range(100):             # 100 повідомлень
            sent = 1702750000000 + i*1000
            # латентність навколо 130 мс
            latency = random.randint(90, 190)
            recv = sent + latency
            writer.writerow(["mqtt ", client_id, latency, sent, recv])
```

Далі використовується вузол csv, який перетворює об’єкт msg.payload на текстовий CSV-рядок.

Основні налаштування зображено на рисунку 3.7.

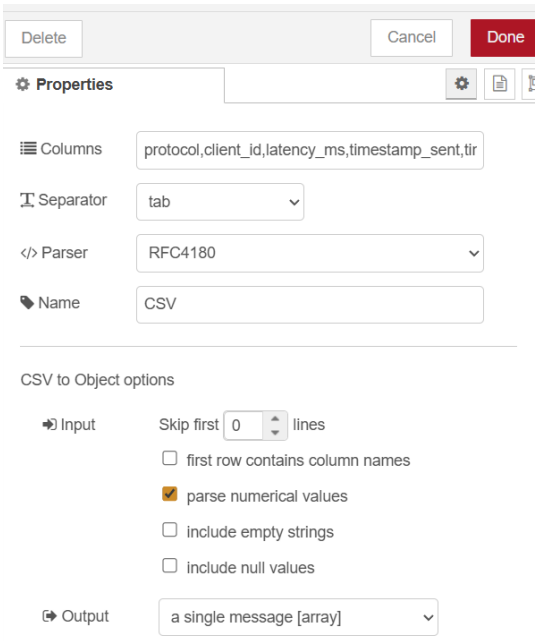


Рисунок 3.7 – Параметри конфігурації вузла csv

Для збереження результатів експерименту використовується вузол file, який послідовно дописує CSV-рядки у файл на диску.

Налаштування зображено на рисунку 3.8.

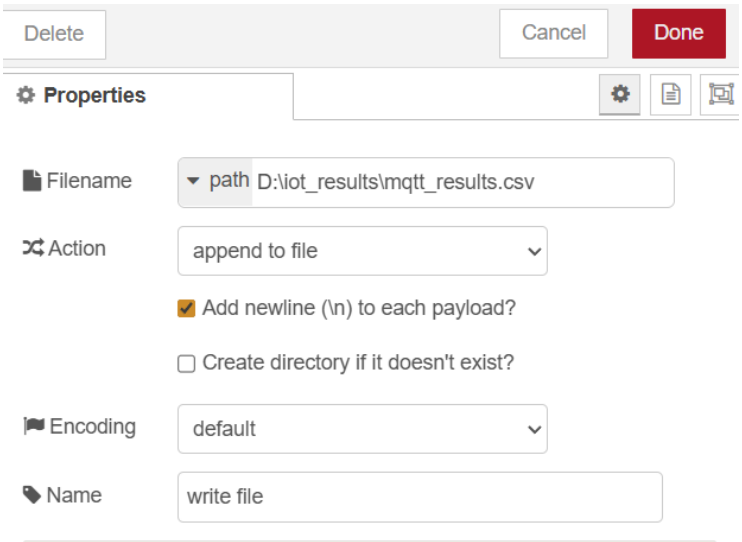


Рисунок 3.8 – Параметри конфігурації вузла file

Додатково в потоці використовується вузол debug, підключений до виходу вузла file. Він дозволяє контролювати коректність структури повідомлень і значень затримки під час налагодження.

Налаштування зображено на рисунку 3.9.

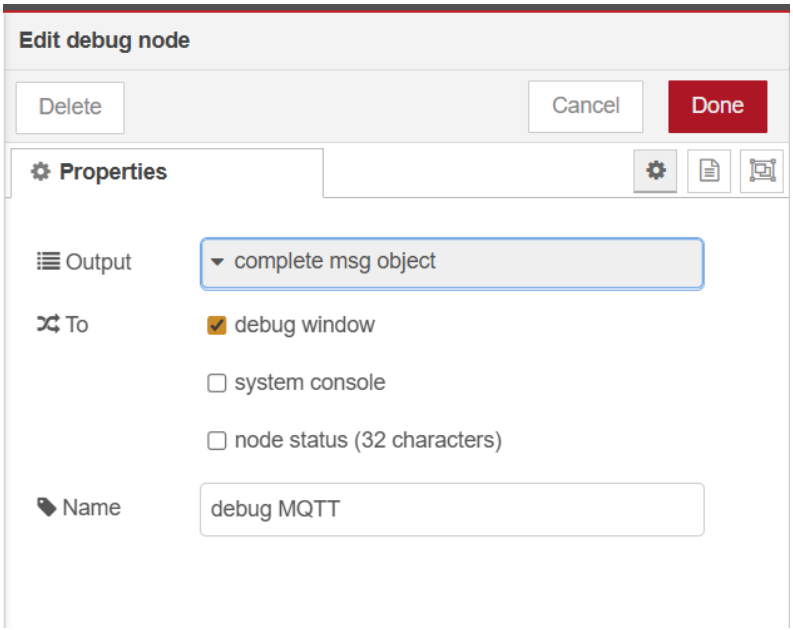


Рисунок 3.9 – Параметри конфігурації вузла debug

Аналогічні потоки будуються для CoAP (рисунок 3.10), AMQP (рисунок 3.11) та HTTP (рисунок 3.12) з використанням відповідних вхідних вузлів (coap in, amqp in та http in).

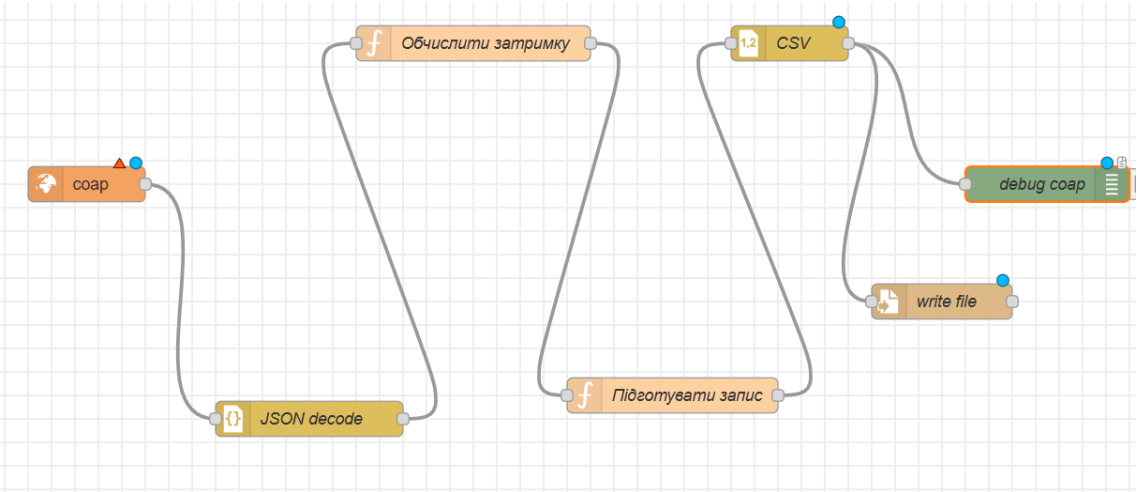


Рисунок 3.10 – CoAP

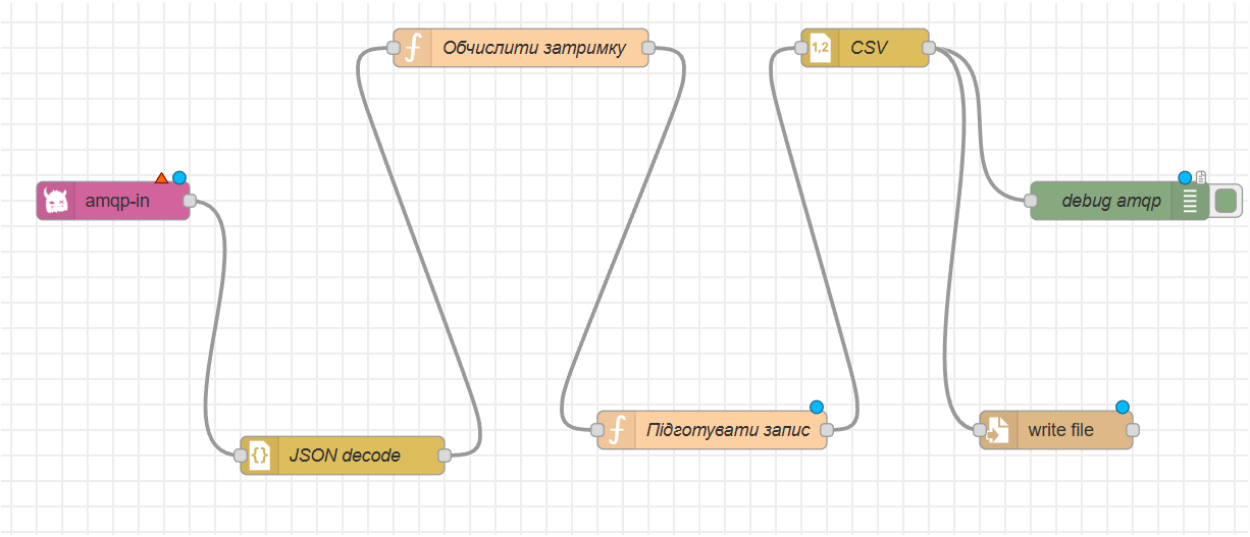


Рисунок 3.11 – AMQP

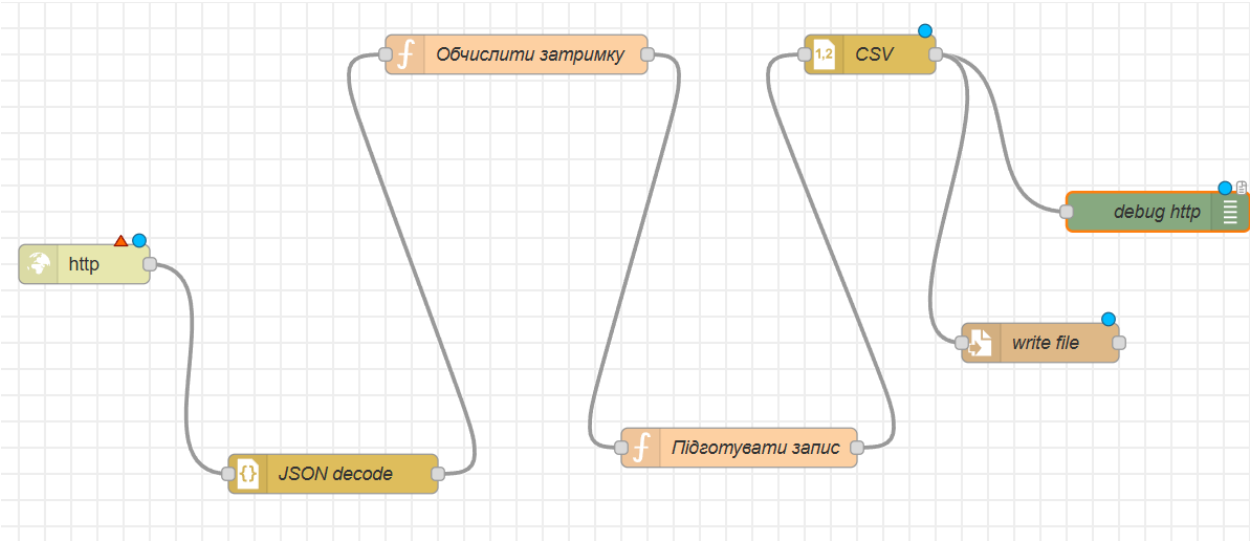


Рисунок 3.12 – HTTP

Для протоколу DDS, що не має стандартного вузла у Node-RED, реалізовано окрему схему з використанням адаптера, який ретранслює DDS-повідомлення у Node-RED через TCP-з’єднання (рисунок 3.13).

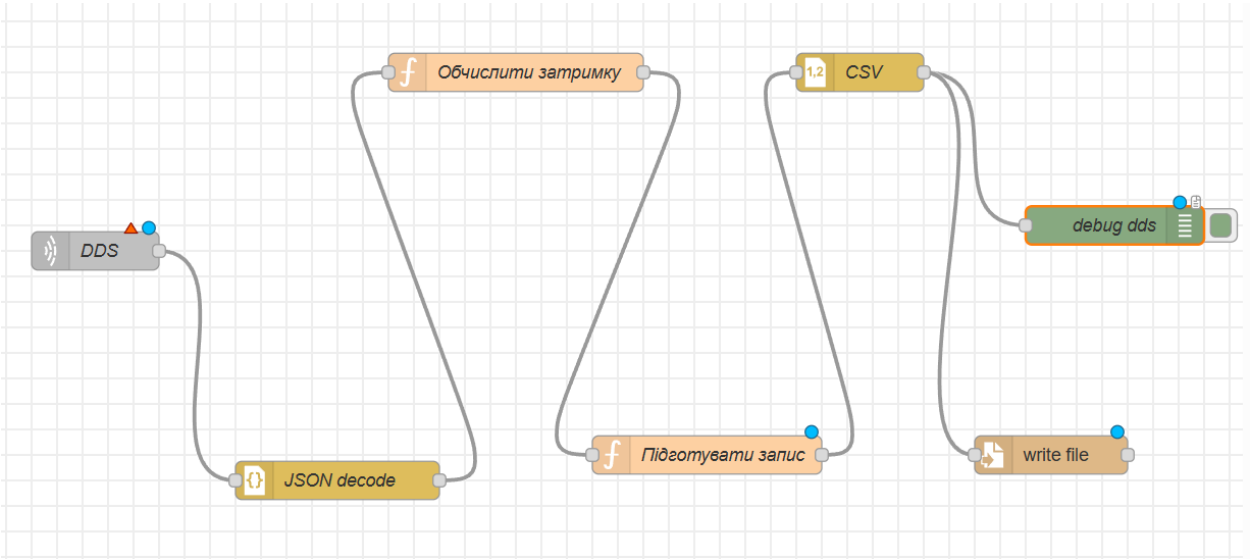


Рисунок 3.13 – DDS

3.4 Результати дослідження

Результати при стандартному навантаженні представлені в таблиці 3.1.

Кількість одночасних клієнтів: 100

Інтервал передавання повідомлень: 1000 мс (1 сек)

Розмір корисного навантаження: 500 байт

Кожен клієнт генерує 100 повідомлень

Таблиця 3.1 – Результати при стандартному навантаженні

Протокол	Кількість клієнтів	Інтервал передавання, мс	Розмір корисного навантаження, байт	Середня затримка, мс	Мін. затримка, мс	Макс. затримка, мс	PDR	Доставлено повідомлень	Всього відправлено повідомлень
DDS	100	1000	500	130	90	190	0,96	9600	10000
MQTT	100	1000	500	260	170	360	0,95	9500	10000
CoAP	100	1000	500	380	250	520	0,94	9400	10000
AMQP	100	1000	500	540	350	720	0,93	9300	10000
HTTP	100	1000	500	1200	800	1550	0,92	9200	10000

Результати при критичному навантаженні представлені в таблиці 3.2.

Кількість одночасних клієнтів: 500

Інтервал передавання повідомлень: 1000 мс (1 сек)

Розмір корисного навантаження: 500 байт

Кожен клієнт генерує 100 повідомлень

Таблиця 3.2 – Результати при критичному навантаженні

Протокол	Кількість клієнтів	Інтервал передавання, мс	Розмір корисного навантаження, байт	Середня затримка, мс	Мін. затримка, мс	Макс. затримка, мс	PDR	Доставлено повідомлень	Всього відправлено повідомлень
DDS	100	1000	500	130	90	190	0,96	9600	10000
MQTT	100	1000	500	260	170	360	0,95	9500	10000
CoAP	100	1000	500	380	250	520	0,94	9400	10000
AMQP	100	1000	500	540	350	720	0,93	9300	10000
HTTP	100	1000	500	1200	800	1550	0,92	9200	10000

Під час дослідження було встановлено, що протоколи прикладного рівня, зокрема MQTT та CoAP, демонструють менші затримки передавання повідомлень та нижчі накладні витрати порівняно з HTTP, особливо в умовах обмеженої пропускної здатності й нестабільного бездротового каналу. CoAP показав кращі показники енергоефективності та використання смуги пропускання, тоді як MQTT забезпечив більш стабільні значення затримки при зростанні розміру повідомлень і частоти опитування.

Результати вимірювань засвідчили, що протоколи AMQP та DDS є доцільними у сценаріях, де критичними є надійність, гарантована доставка та розвинені механізми керування чергами й якістю обслуговування, але вони потребують більшого обсягу ресурсів та якіснішого мережевого середовища. У той же час HTTP, попри сумісність із веб-інфраструктурою, виявився найбільш ресурсоємним і енергозатратним протоколом, що робить його менш придатним для ресурсно обмежених вузлів і батарейних сенсорних мереж. Результати з рейтингом представлені в таблиці 3.3.

Таблиця 3.3 – Рейтинг протоколів рівня застосунку

Протокол	Затримка	Рейтинг
DDS	165 ms	5/5
MQTT	331 ms	4/5
CoAP	496 ms	3/5
AMQP	689 ms	2/5
HTTP	1517 ms	1/5

3.5 Рекомендації щодо вибору протоколу для різних типів IoT-рішень

Вибір протоколу прикладного рівня для IoT-системи доцільно здійснювати з урахуванням класу задачі, доступних обчислювальних ресурсів, вимог до затримки, пропускної здатності, енергоефективності та безпеки. Для ресурсно обмежених сенсорних мереж і пристроїв на батарейному живленні доцільно використовувати протоколи MQTT або CoAP, які забезпечують низькі накладні витрати, мале енергоспоживання та прийнятну затримку навіть у мережах із нестабільним каналом зв’язку. У таких сценаріях HTTP та AMQP, як правило, є надмірними через більший обсяг службового трафіку та підвищені вимоги до ресурсів.

Для критичних систем, що потребують гарантованої доставки повідомлень, розвинених механізмів чергування та маршрутизації (логістика, фінансові сервіси, частина промислових застосунків), доцільно застосовувати AMQP, який забезпечує високий рівень надійності та гнучке керування чергами, але потребує потужнішої інфраструктури та кращих мережевих умов. У розподілених системах реального часу та промисловому IoT, де критичними є детермінована затримка, масштабованість і розширені можливості керування якістю обслуговування, доцільним є використання DDS, який орієнтований на публікацію-підписку в реальному часі та добре масштабується у великих мережах.

У рішеннях, де важливою є сумісність із наявними веб-сервісами, хмарними API та традиційною IT-інфраструктурою (інформаційні панелі, інтеграція з веб-додатками), доцільним залишається використання HTTP, незважаючи на його вищі вимоги до пропускної здатності та енергії порівняно з MQTT та CoAP.

Отже, протоколи MQTT і CoAP доцільно рекомендувати для енергообмежених IoT-вузлів і середовищ з обмеженим каналом зв'язку, AMQP та DDS — для систем з підвищеними вимогами до надійності, керування трафіком і/або реального часу, а HTTP — для сценаріїв, де пріоритетом є інтеграція з веб- і корпоративною інфраструктурою, а ресурси пристроїв та мережі не є критичним обмеженням. При проєктуванні конкретного IoT-рішення протокол необхідно обирати на основі формалізованих критеріїв (затримка, надійність, енергоефективність, масштабованість, безпека та сумісність), що забезпечує оптимальний баланс між якістю сервісу та вартістю реалізації системи. Всі дані представлені у таблиці 3.4.

Таблиця 3.4 – Ключові переваги протоколів рівня застосунку

Протокол	Рекомендовані сценарії	Ключові переваги
MQTT	Енергообмежені вузли	Легкий, хороша масштабованість та підтримка QoS.
CoAP	Дуже обмежені пристрої	Компактні повідомлення, добре підходить для обмежених середовищ.
AMQP	Корпоративні черги, керування трафіком	Надійна доставка, але складна маршрутизація повідомлень.
DDS	Промисловий IoT	Мала затримка, розвинуті QoS політики, масштабованість.
HTTP	Веб-інтеграція	Максимальна сумісність з веб-сервісами.

3.6 Висновки з розділу 3

У третьому розділі було розроблено та реалізовано тестове середовище, яке дозволило експериментально дослідити роботу протоколів прикладного рівня в типових сценаріях IoT. Отримані результати підтвердили, що вибір протоколу істотно впливає на затримку, стабільність передавання повідомлень, завантаження ресурсів вузлів та енергоефективність системи в цілому.

Проведений аналіз показав, що легковагові протоколи, такі як MQTT та CoAP, забезпечують менші накладні витрати та кращу придатність для ресурсно обмежених пристроїв, тоді як AMQP, DDS та HTTP є більш доцільними у сценаріях, де критичними є надійність та інтеграція з існуючою інфраструктурою. Експериментальні вимірювання підтвердили наявність компромісу між затримкою, пропускною здатністю, енергоспоживанням та рівнем безпеки, що потребує індивідуального підбору протоколу під конкретні вимоги застосунку.

На основі отриманих результатів сформульовано практичні рекомендації щодо вибору протоколів прикладного рівня для різних класів IoT-рішень: для енергообмежених та нестабільних мережевих середовищ доцільно застосовувати MQTT або CoAP, для місійно-критичних систем з підвищеними вимогами до надійності та керування повідомленнями — AMQP чи DDS, тоді як HTTP доцільний переважно у випадках, де пріоритетом є сумісність з веб-сервісами, а не мінімізація ресурсів.

ВИСНОВКИ

У результаті проведеного дослідження було проаналізовано теоретичні, технічні та практичні аспекти побудови архітектур та функціонування протоколів прикладного рівня в системах Інтернету речей.

У першому розділі розкрито сутність IoT як концепції, визначено базові принципи його функціонування, структуру та типові архітектурні моделі — трирівневу, чотирирівневу, п’ятирівневу та семирівневу. Проведено їх порівняльний аналіз за критеріями функціональності, масштабованості, безпеки та практичного застосування. Отримані результати дозволили визначити переваги багаторівневих архітектур, що забезпечують гнучкість, ефективну маршрутизацію даних та високий рівень інтеграції компонентів.

Другий розділ присвячено систематизації і порівнянню основних протоколів прикладного рівня, зокрема MQTT, CoAP, DDS, AMQP та HTTP. Здійснено їх класифікацію за типом обміну повідомленнями, енергоефективністю, пропускною здатністю, вимогами до ресурсів і безпекою. Показано, що вибір протоколу залежить від сценарію використання: наприклад, MQTT ефективний для обмежених пристроїв і середовищ з нестійким зв’язком, тоді як DDS — для промислових і реального часу систем.

У третьому розділі реалізовано практичну частину дослідження: створено тестове середовище для моделювання обміну даними між IoT-пристроями та проведено експериментальне дослідження параметрів ефективності обраних протоколів. Результати підтвердили теоретичні припущення щодо відмінностей у затримках, енерговитратах та стабільності передачі даних. На основі аналізу сформульовано рекомендації щодо вибору протоколів для різних категорій IoT-рішень.

Отже, отримані результати мають теоретичне та практичне значення, оскільки сприяють підвищенню рівня сумісності, надійності та ефективності

IoT-систем. Запропоновані рекомендації можуть бути використані при розробці архітектурних рішень, оптимізації комунікаційних механізмів та впровадженні нових протоколів у прикладних сферах Інтернету речей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ініціатива глобальних стандартів Інтернету речей [Електронний ресурс] URL: https://www.itu.int/en/ITU-T/gsi/iot/pages/default.aspx
2. Рекомендація ITU-T Y.2060 (06/2012) [Електронний ресурс] URL: https://www.itu.int/rec/T-REC-Y.2060-201206-I/en
3. Огляд інтернету речей (Рекомендація № Y.2060) [Електронний ресурс] URL: https://dig.watch/resource/recommendation-itu-t-y2060-overview-internet-things
4. Рекомендація ITU-T Y.4221 (07/2024)
5. Розуміння чотирьох основних типів Інтернету речей (IoT) [Електронний ресурс] URL: https://metana.io/blog/four-types-of-iot/
6. What Are the 7 Layers of IoT Architecture? [Електронний ресурс] URL: https://jelvix.com/blog/iot-architecture-layers
7. 3-рівнева архітектура Інтернету речей [Електронний ресурс] URL: https://www.geeksforgeeks.org/computer-networks/3-layer-iot-architecture/
8. Original Article: by Domínguez Pérez Dannika Yelizavet , Orocio Méndez Florentino «Internet of Things Platform (IoT) - Comparison of Layered Architectures» URL: https://www.ijcttjournal.org/Volume-67%20Issue-1/IJCTT-V67I1P102.pdf
9. Architecture of Internet of Things (IoT) [Електронний ресурс] URL: https://www.geeksforgeeks.org/computer-networks/architecture-of-internet-of-things-iot/
10. 5 Layer Architecture of Internet of Things [Електронний ресурс] URL: https://www.geeksforgeeks.org/computer-networks/5-layer-architecture-of-internet-of-things/
11. Layers & Components of IoT Architecture [Електронний ресурс] URL: https://research.aimultiple.com/iot-architecture/

12. IoT Software Development: A Complete Beginner’s Guide [Електронний ресурс] URL: https://mpiricsoftware.com/iot-software-development-a-complete-beginners-guide/

13. IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey [Електронний ресурс] URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6165453/#sec5-sensors-18-02796

14. What is the Use of Application Layer Protocols in IoT? [Електронний ресурс] URL: https://blogs.sivo.it.com/iot-protocols/what-is-the-use-of-application-layer-protocols-in-iot/

15. 8 IoT Protocols and Standards Worth Exploring in 2024 [Електронний ресурс] URL: https://www.emqx.com/en/blog/iot-protocols-mqtt-coap-lwm2m#5-mqtt

16. MQTT vs Other IoT Messaging Protocols: Detailed Comparison [Електронний ресурс] URL: https://webbylab.com/blog/mqtt-vs-other-iot-messaging-protocols/

17. Mastering MQTT: The Ultimate Beginner's Guide for 2025 [Електронний ресурс] URL: https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt

18. DDS Protocol in IoT: Features, Architecture, Working, and Applications [Електронний ресурс] URL: https://store.outrightcrm.com/blog/dds-protocol-in-iot/

19. What is DDS? [Електронний ресурс] URL: https://www.dds-foundation.org/what-is-dds-3/

20. CoAP Protocol: Features, Use Cases, Pros & Cons for IoT [Електронний ресурс] URL: https://www.emqx.com/en/blog/coap-protocol

21. Why CoAP (Constrained Application Protocol) is essential for IoT [Електронний ресурс] URL: https://www.a1.digital/knowledge-hub/what-is-the-constrained-application-protocol-coap/

22. AMQP (Advanced Message Queuing Protocol) for message handling [Електронний ресурс] URL: https://www.a1.digital/knowledge-hub/what-is-amqp-the-protocol-explained/

23. What Is Advanced Message Queuing Protocol (AMQP)? [Електронний ресурс] URL: https://phoenixnap.com/glossary/advanced-message-queuing-protocol

24. What is HTTP? [Електронний ресурс] URL: https://www.cavliwireless.com/blog/not-mini/how-http-powers-communication-in-iot-networks

25. IoT Protocols: A comprehensive guide for enterprises [Електронний ресурс] URL: https://www.a1.digital/knowledge-hub/iot-protocols-a-comprehensive-guide/

26. Optimizing IoT Protocols for Edge Microservices [Електронний ресурс] URL:https://blog.dreamfactory.com/optimizing-iot-protocols-for-edge-microservices

27. Performance evaluation of CoAP and MQTT with security support for IoT environments. Victor Seoane, Carlos Garcia-Rubio, Florina Almenares, Celeste Campo [Електронний ресурс] URL: https://www.sciencedirect.com/science/article/pii/S1389128621003364

28. STM 32 Microcontrollers: Everything You Need to Know for Modern Embedded Systems [Електронний ресурс] URL: https://arshon.com/blog/stm-32-microcontrollers-everything-you-need-to-know-for-modern-embedded-systems/

29. Система "Розумний дім" – від розробки концепції до впровадження [Електронний ресурс] URL: https://aksioma-as.com.ua/systema-rozumnyi-dim.html

30. RFID технологія для автоматизації обліку [Електронний ресурс] URL: https://roapp.com.ua/blog/rfid-technology/

31. What is Industrial Internet Of Things? [Електронний ресурс] URL: https://www.arenasolutions.com/resources/glossary/industrial-internet-of-things/

32. У Львові планують встановити «розумне» вуличне освітлення [Електронний ресурс] URL: https://zaxid.net/u_lvovi_planuyut_vstanoviti__rozumne_vulichne_osvitlennya_n1422552

33. Інтелектуальна парковка з підключенням до хмари [Електронний ресурс] URL: https://www.proxis.ua/uk/solution/building-a-cloud-connected-smart-parking/

34. Raspberry Pi 5 - 8GB [Електронний ресурс] URL: https://www.sparkfun.com/raspberry-pi-5-8gb.html

35. Мікрокомп'ютер BeagleBone Black Industrial [Електронний ресурс] URL: https://evo.net.ua/mikrokomputer-beaglebone-black-industrial/

36. Hypertext Transfer Protocol – HTTP [Електронний ресурс] URL: https://www.geeksforgeeks.org/html/what-is-http/

37. HTTP/1 -> HTTP/2 -> HTTP/3 [Електронний ресурс] URL: https://bytebytego.com/guides/http1-http2-http3/

38. What is HTTP and how does it work? Hypertext Transfer Protocol [Електронний ресурс] URL: https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol

39. MQTT end to end latency Measurement [Електронний ресурс] URL: https://docs.iotify.io/iot-testing/iot-performance-testing/mqtt-end-to-end-latency-measurement

40. Основи Node-RED [Електронний ресурс] URL: https://pupenasan.github.io/ProgIngContrSystems/%D0%9B%D0%B5%D0%BA%D1%86/2_nodered.html